





# NI myRIO

*Design Real Systems, Fast*



# Agenda



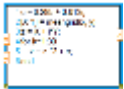
- Overview of NI myRIO



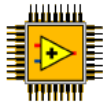
- Introduction to LabVIEW



- Introduction to LabVIEW Real-Time



- LabVIEW Programming Interfaces



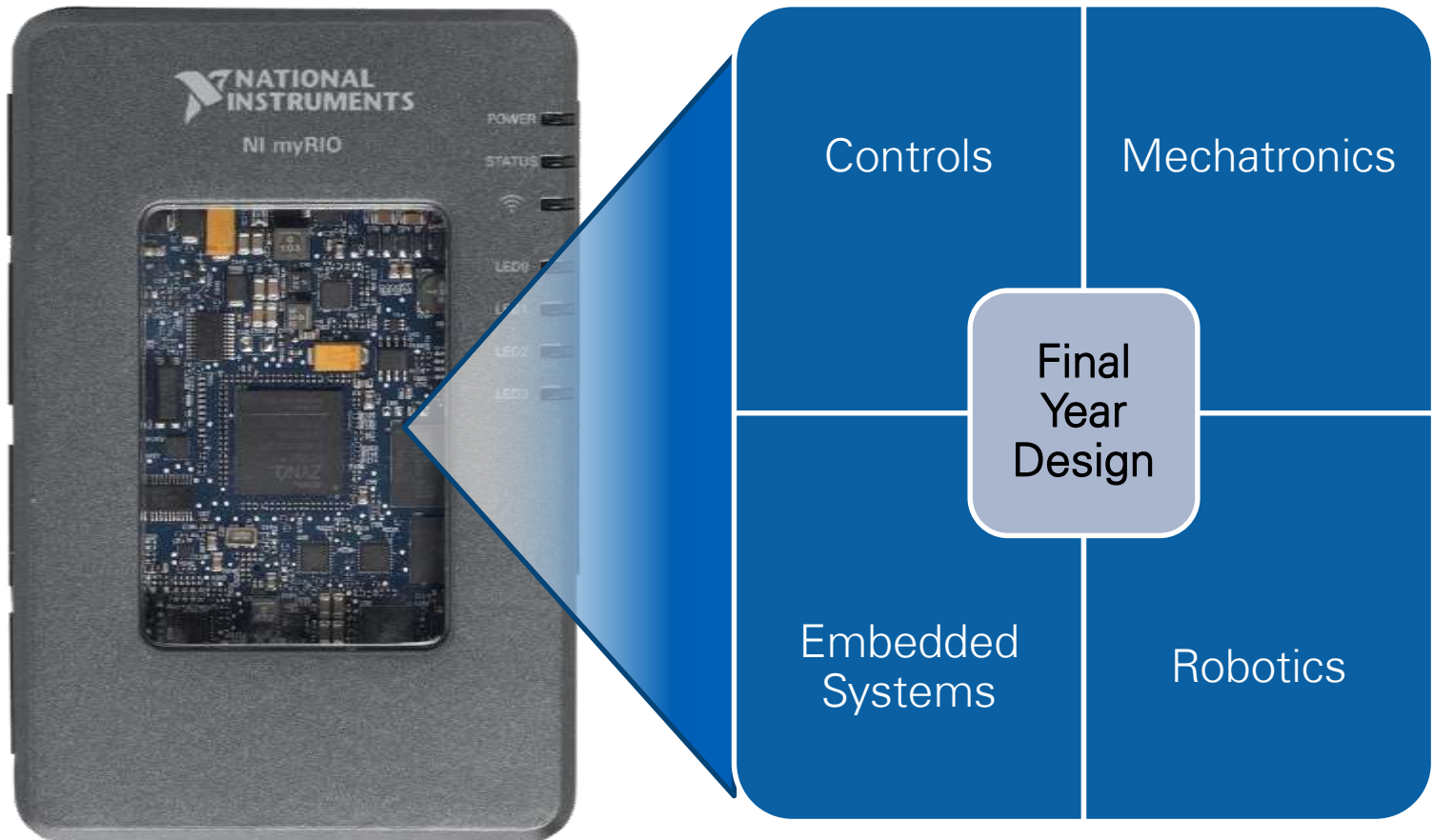
- Introduction to LabVIEW FPGA



- Resources and Next Steps

# Overview of NI myRIO

# From Core Concepts to System Design



# NI myRIO

## Enclosed

Feature Rich

Protective Casing

Ecosystem



## Board-Only

Optimized for Cost

- No Wi-Fi
- No myDAQ connector

Integrate into a Larger Project



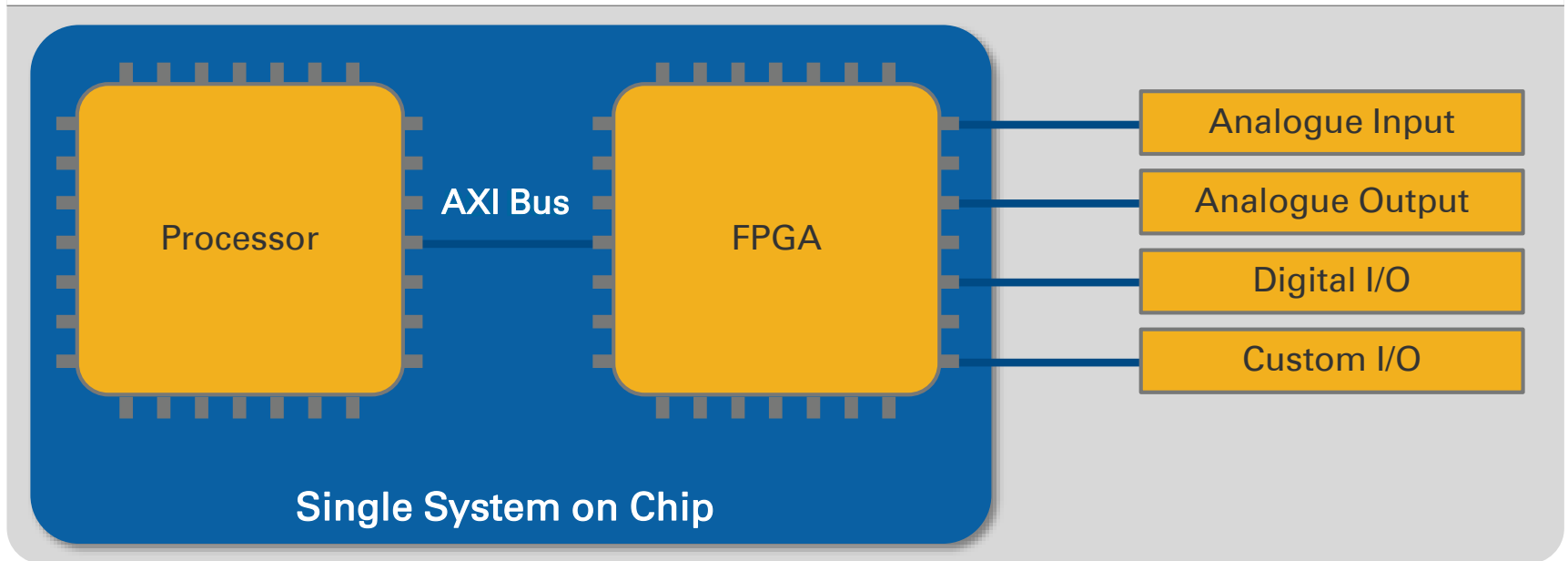
# NI myRIO Product Overview: Front View



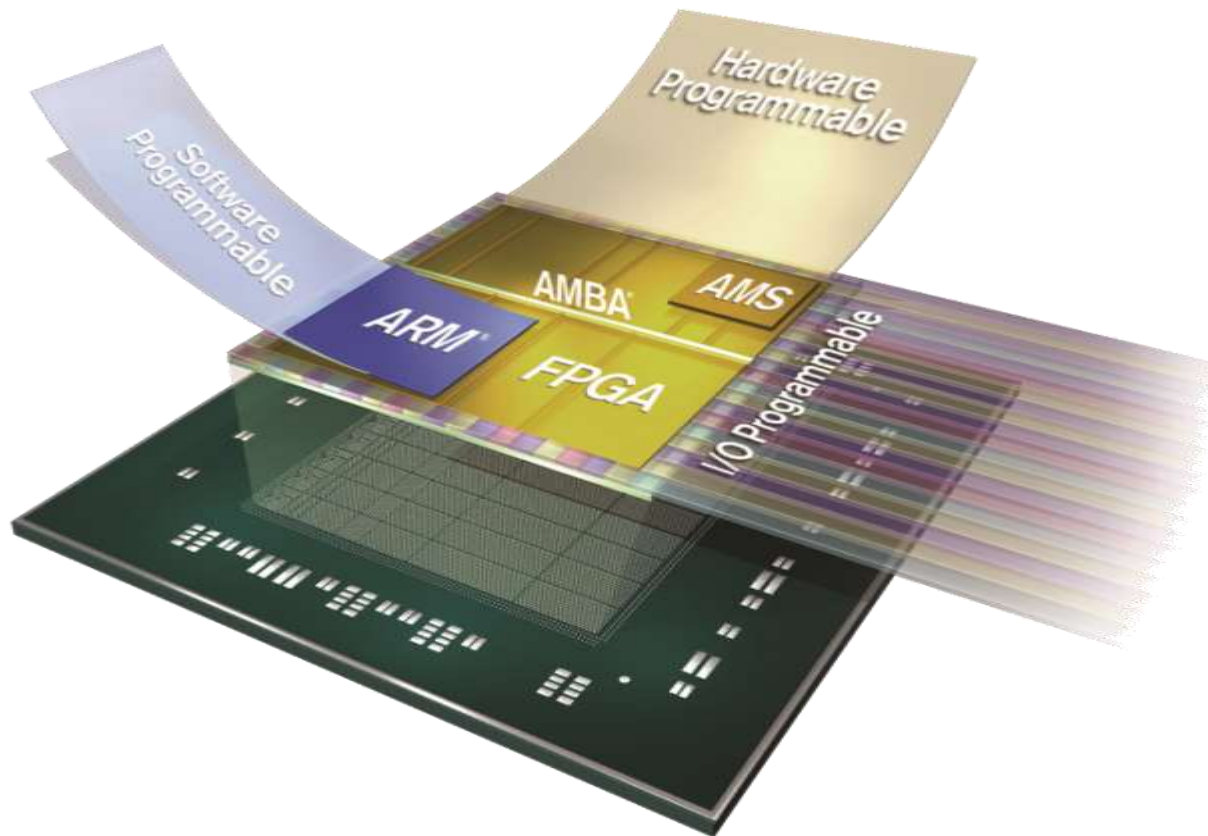
XILINX Zynq SoC

# What is Zynq?

## Traditional Implementation







667 MHz Dual-Core ARM Cortex-A9 processor  
28K Logic Cells (Artix-7)  
80 DSP slices, 16 DMA channels  
92 Billion calculations per second

# Why Zynq Matters in Education



- Smaller Size, Lower Power
- 667 MHz Dual-Core ARM Cortex-A9 Processor
- Artix-7 FPGA, 28k logic cells
- 16 DMA Channels

# Why Zynq Really Matters in Education



## Leading Industry Grade Technology



The same technology is used in our latest industry and research ready Compact RIO systems





# CompactRIO Applications



Controlling a Robotic Manipulator for Nuclear Decommissioning



Tuning Aston Martin Engines for Endurance Races



Plasma control in the world's first bench top Tokamak



# CompactRIO Applications



Controlling 70-Ton Robotic Gripper Arms for Offshore Wind Turbine Construction

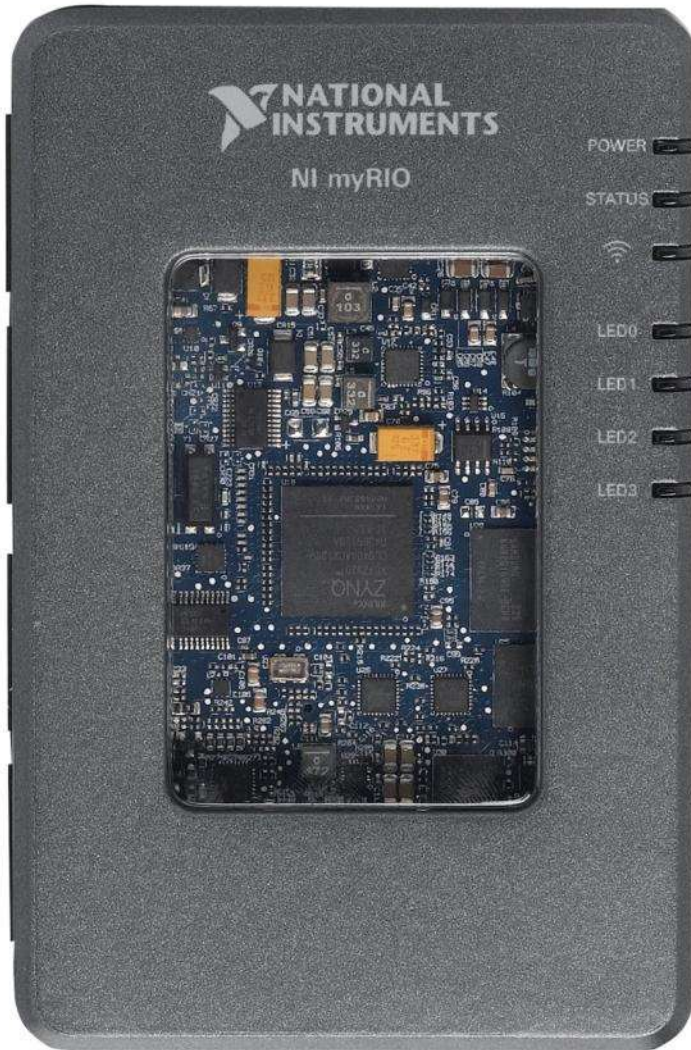


Measuring Biomechanical Stresses in Rugby Scrummaging



Control and Monitor Community Generation Sources in Canada's Smart Grid

# NI myRIO Product Overview: Front View



User Defined LEDs



# Back View

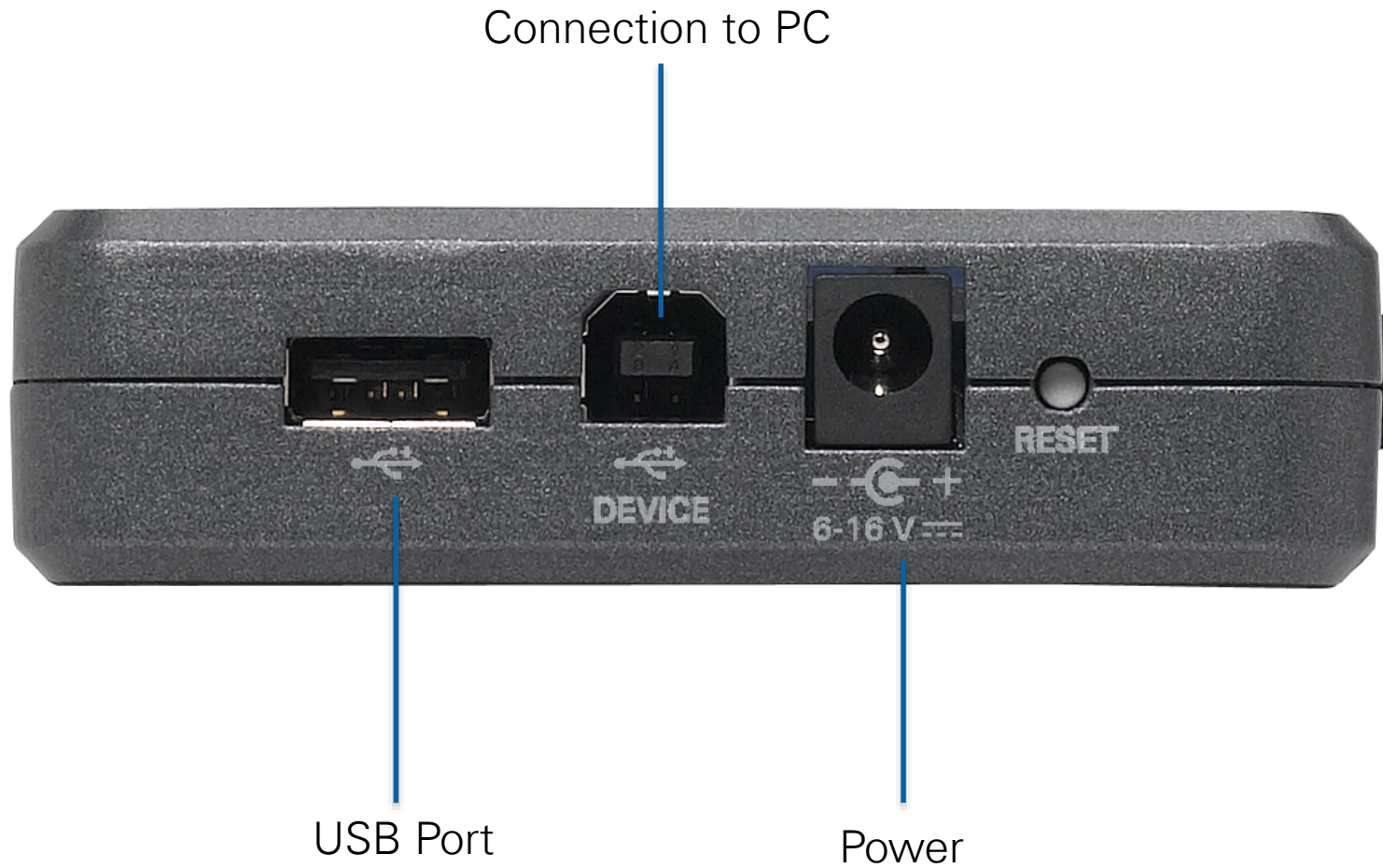


Built-in Accelerometer

Mounting Holes

Getting Started

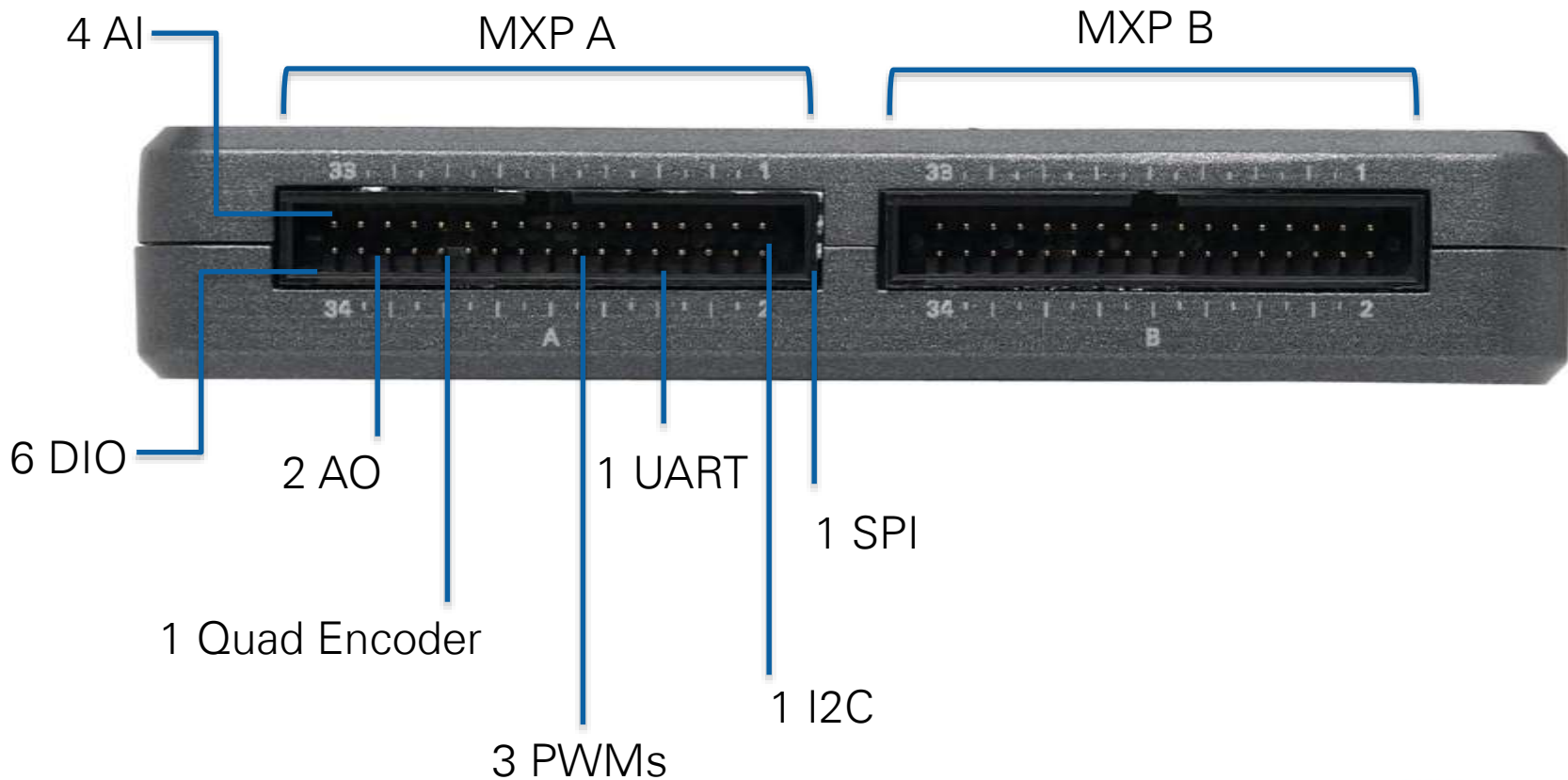
# Top View



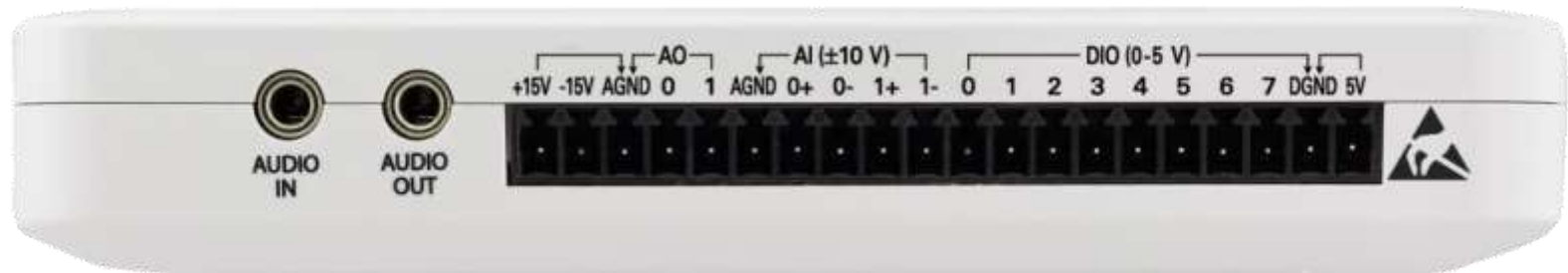
# NI myRIO Expansion Port (MXP)



Identical Connectors



# NI miniSystems Port (MSP)

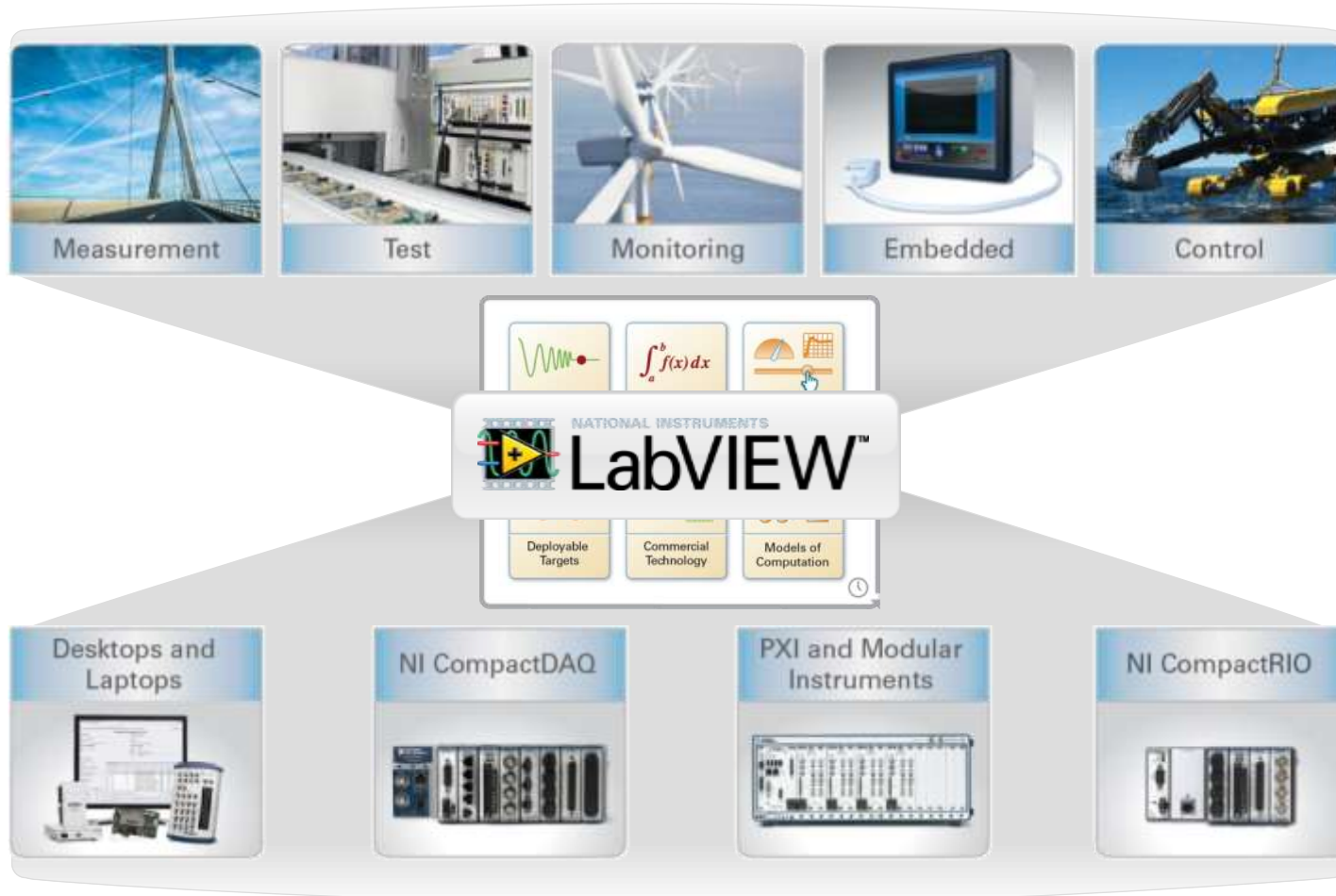


# Introduction to NI LabVIEW



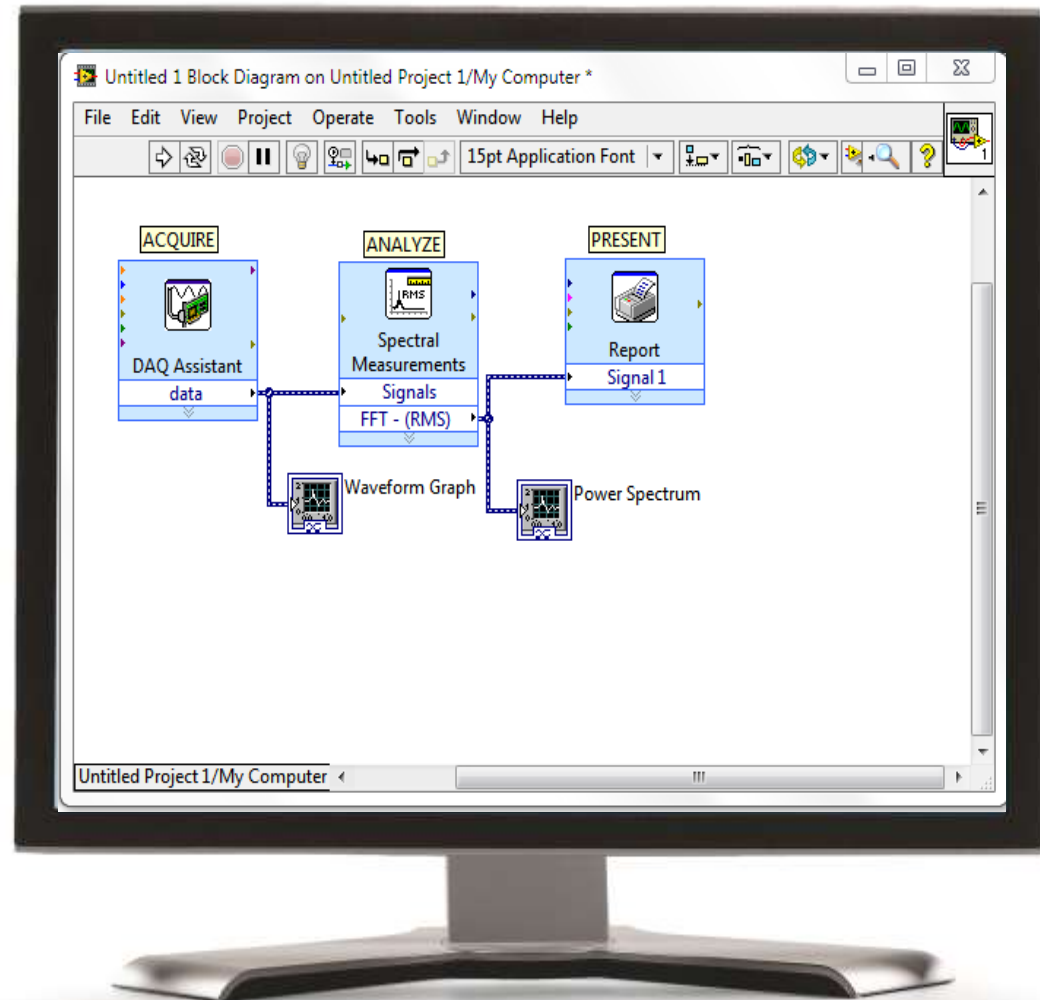
# Graphical System Design

A platform-based approach for measurement and control

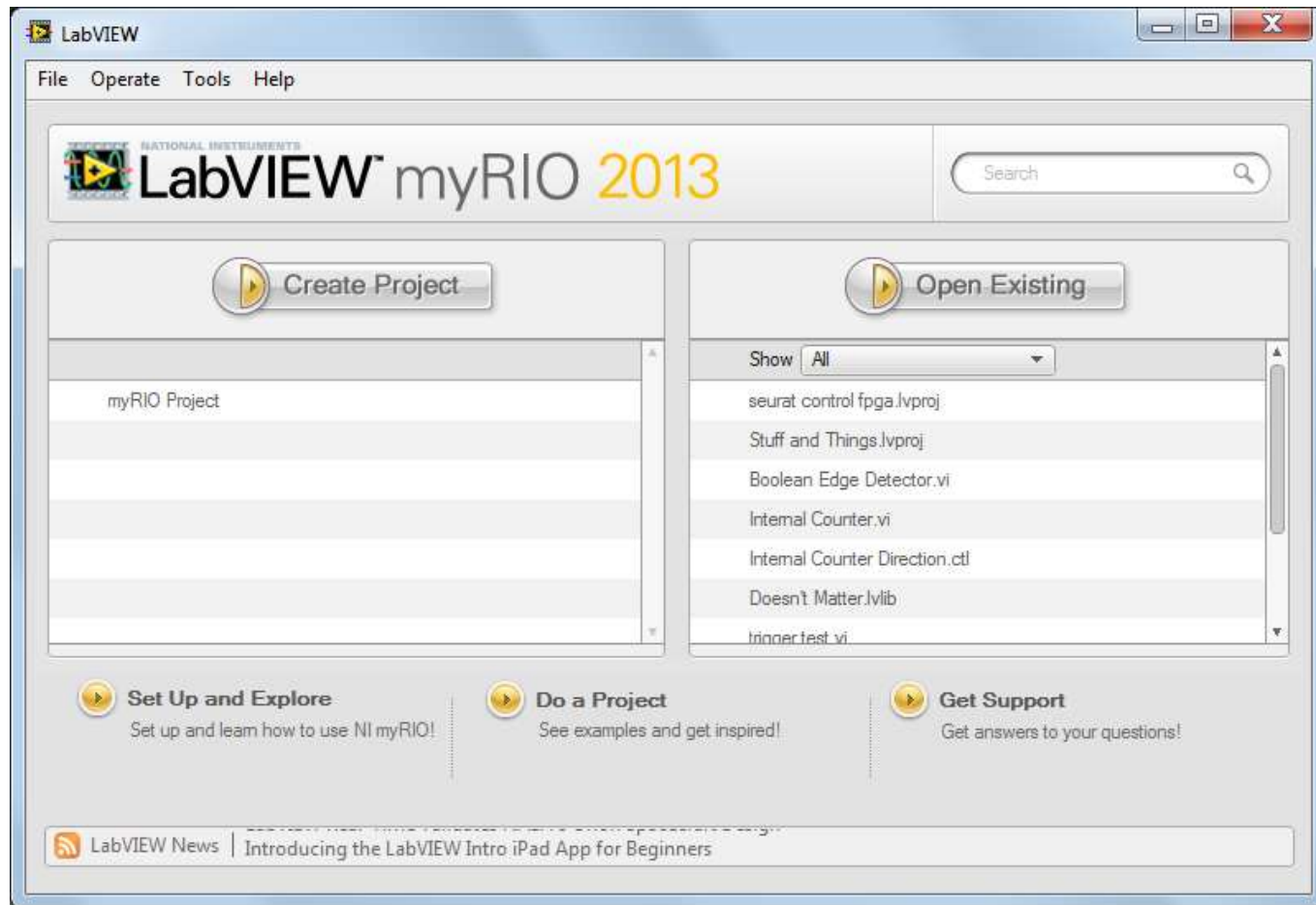




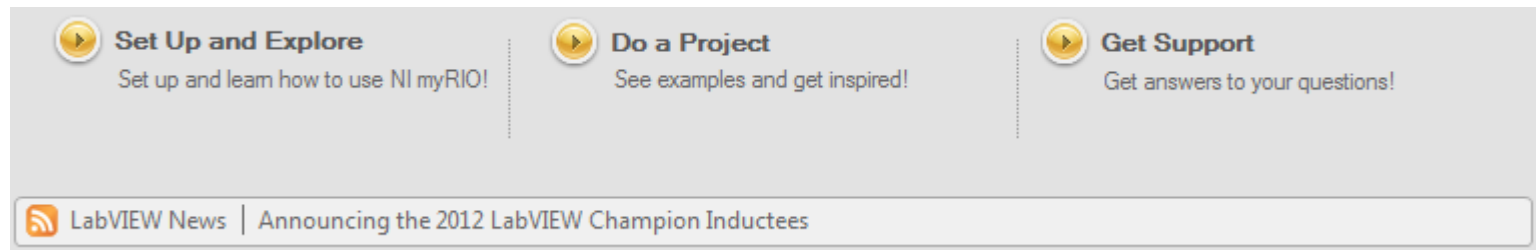
# Data Flow



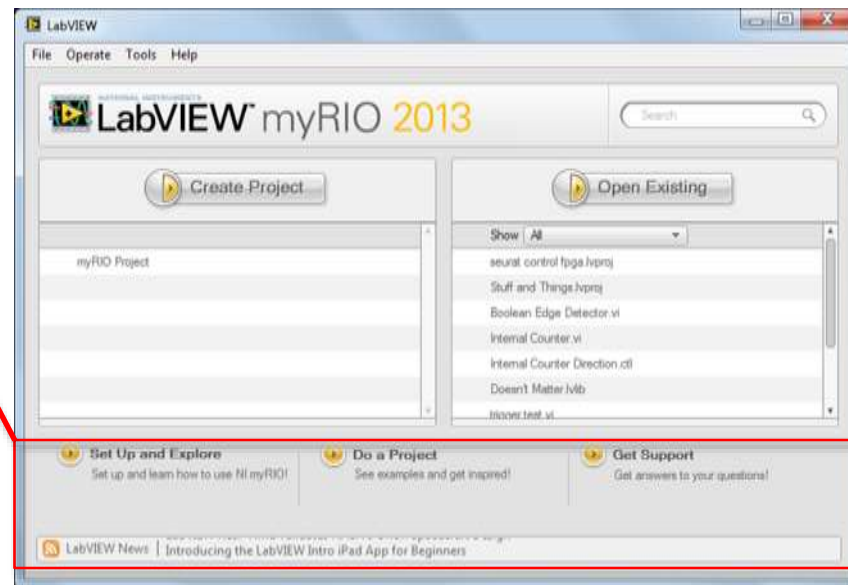
# LabVIEW Getting Started Window



# LabVIEW Getting Started Window



Additional support, tutorials, and explanations can all be found using the links here. These are specifically tailored to NI myRIO users.



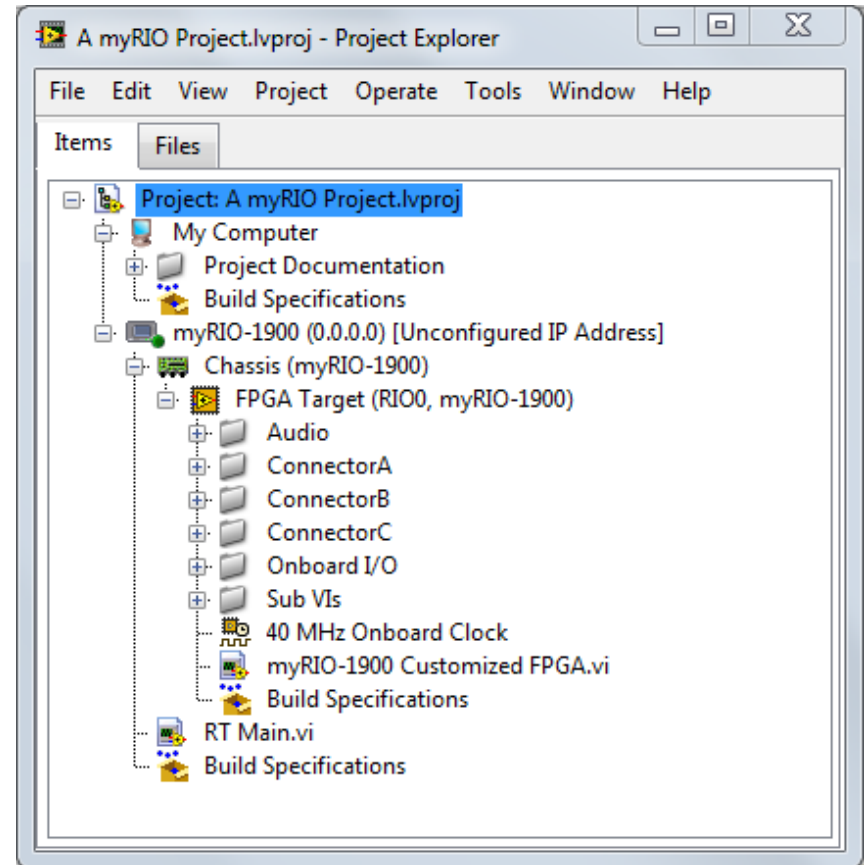
# Create a Project

- Click the **Create Project** button
- Select **Blank Project**.
- Click **Finish**.
- To save the project:
  - **File >> Save**
  - Select the desired directory and choose a meaningful name.
  - Remember, two LabVIEW projects cannot share the same directory.



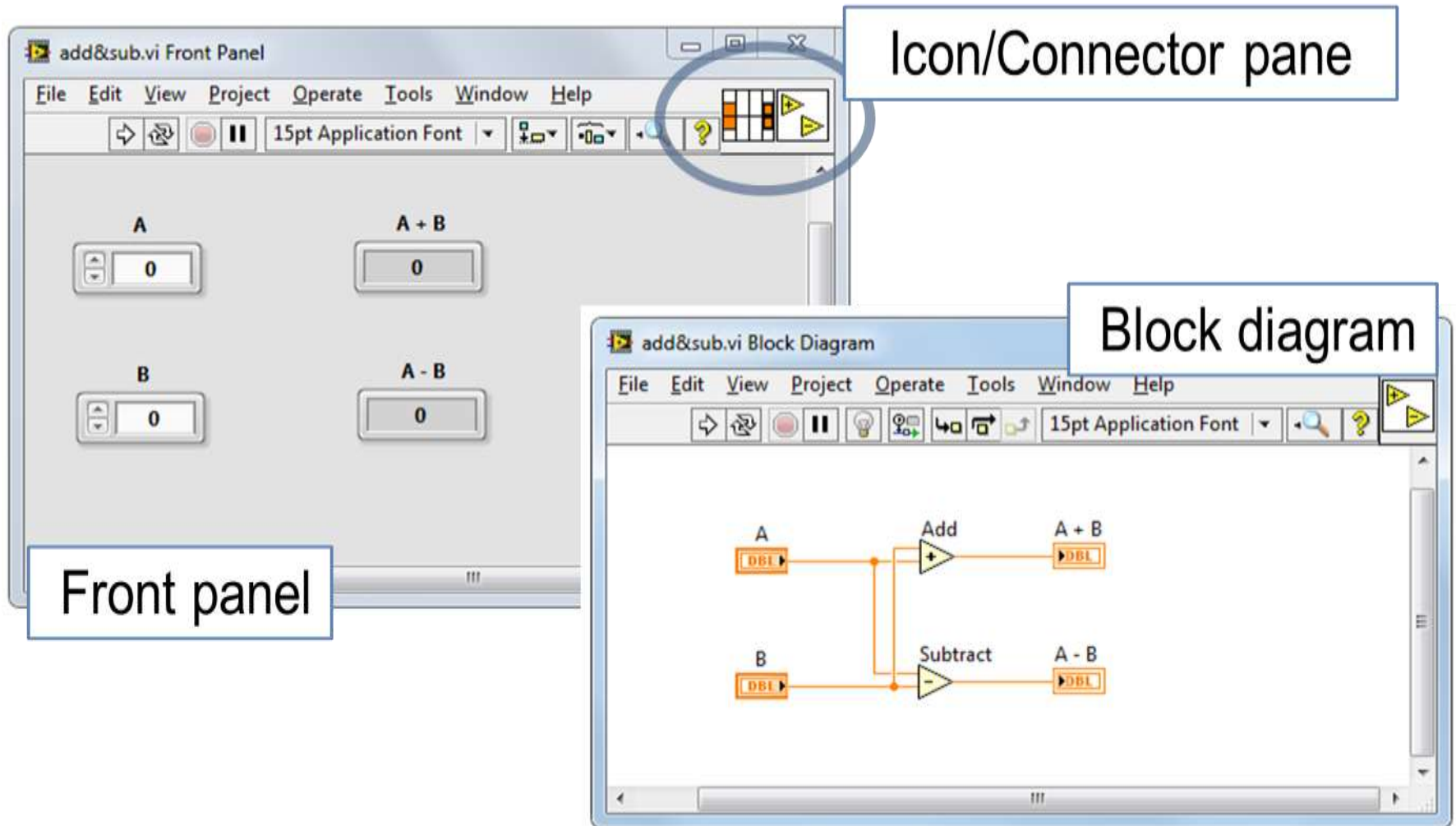
# Project Explorer

- Find, access, and organize project files
- Deploy or download files to targets
- Manage code for build options
  - Executables, installers, and zip files



# Parts of a VI

VIs have three main components:

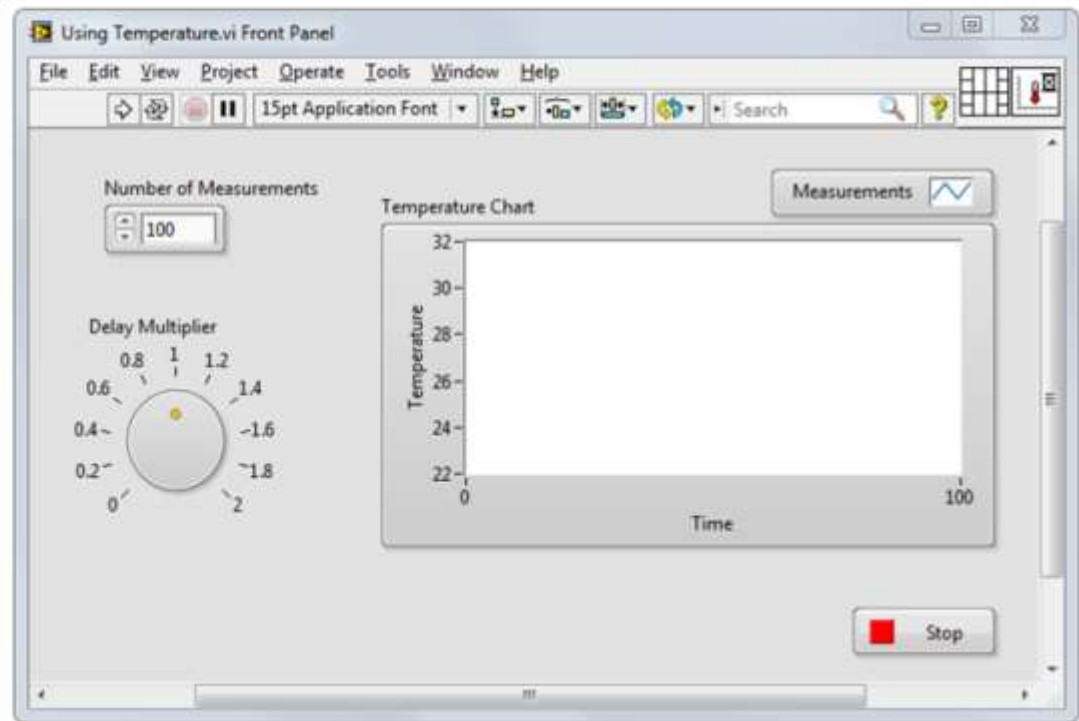




# Parts of a VI – Front Panel

## Front Panel – User interface for the VI

The front panel is constructed using controls (inputs) and indicators (outputs).

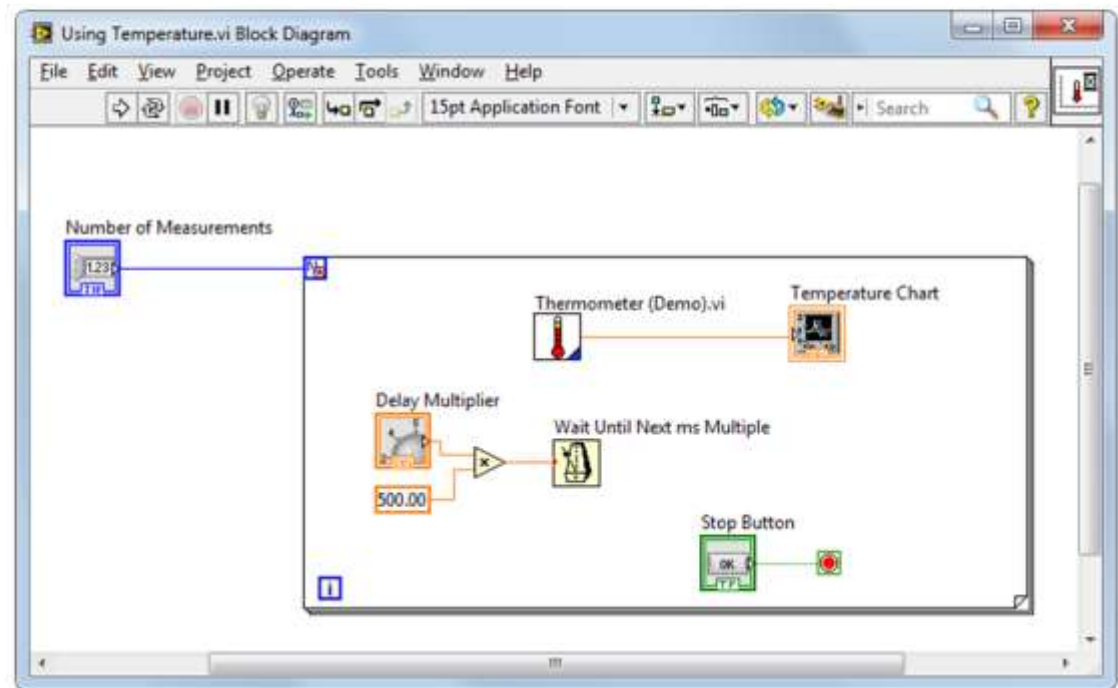


# Parts of a VI – Block Diagram

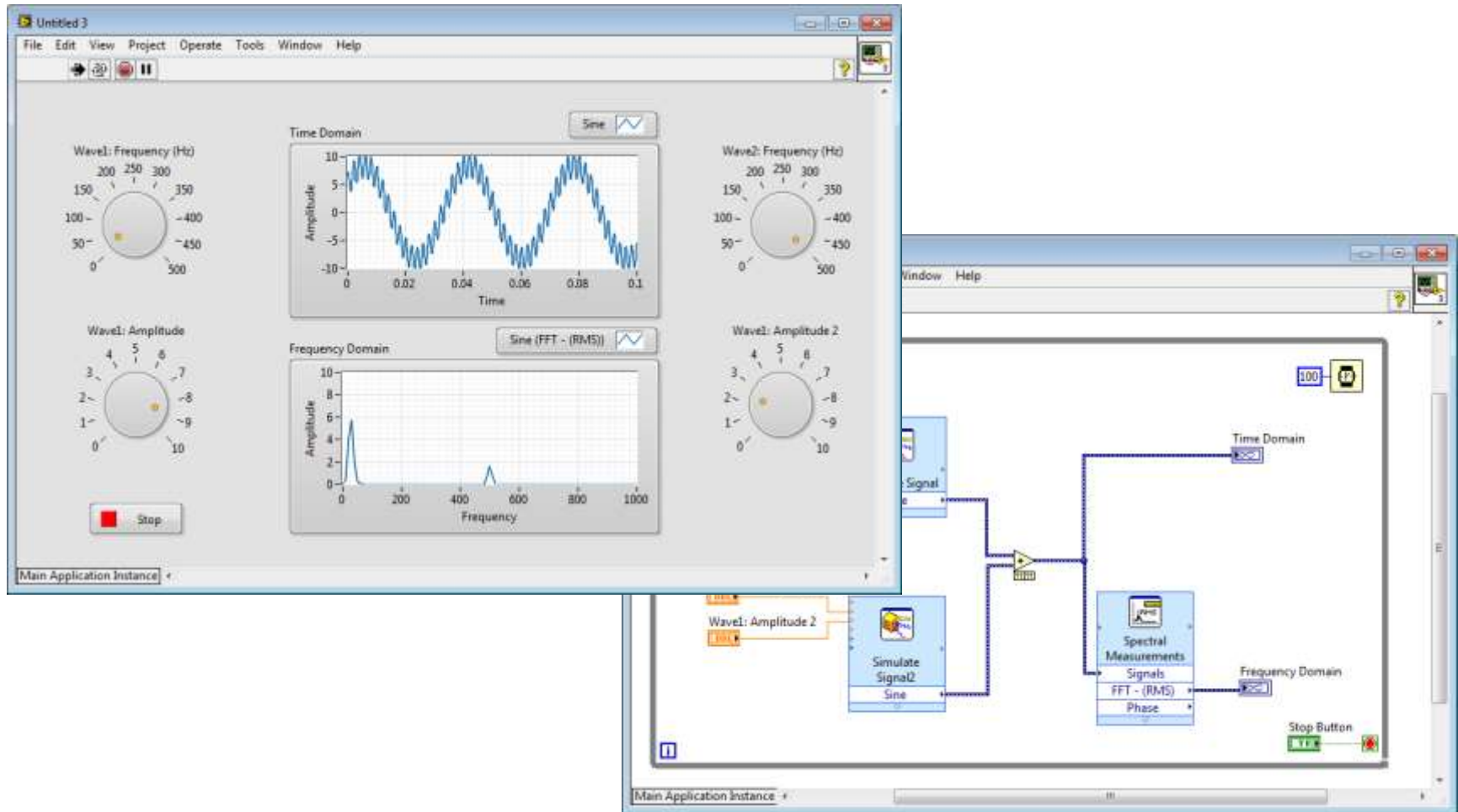
**Block Diagram** – Contains the graphical source code

Front panel object appear as terminals on the block diagram.

This is where “programming” is done in LabVIEW.



# Creating a LabVIEW Application



# Introduction to LabVIEW Real-Time

# What is Real-Time?

- Real-time **does not** always mean real fast
- Real-time means **absolute reliability**
- Real-time systems have timing constraints that must be met to avoid failure
- Determinism is the timing reliability of the system





# Critical Applications to Consider

Event Response



Closed-Loop Control



Critical Tests

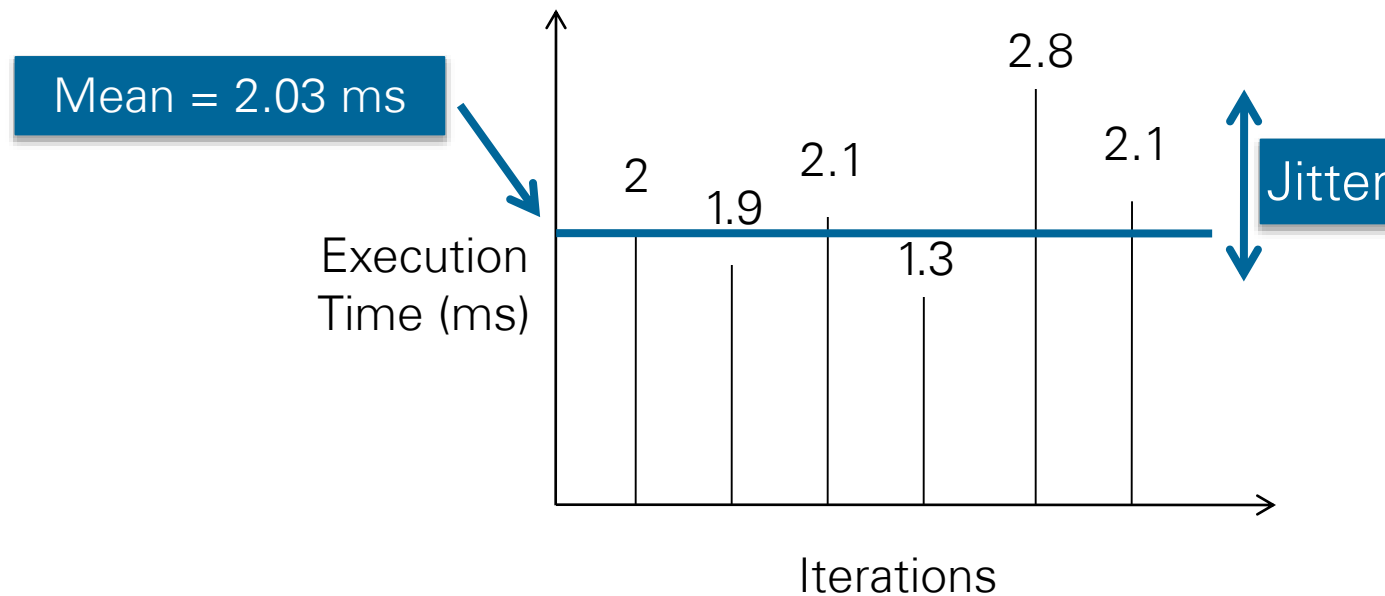


# When General Purpose OSs Fall Short

- Design for fairness and user responsiveness vs. strictly prioritizing tasks
- Focus on multitasking instead of maximum reliability / uptime
- Not the result of bad products, only certain design goals

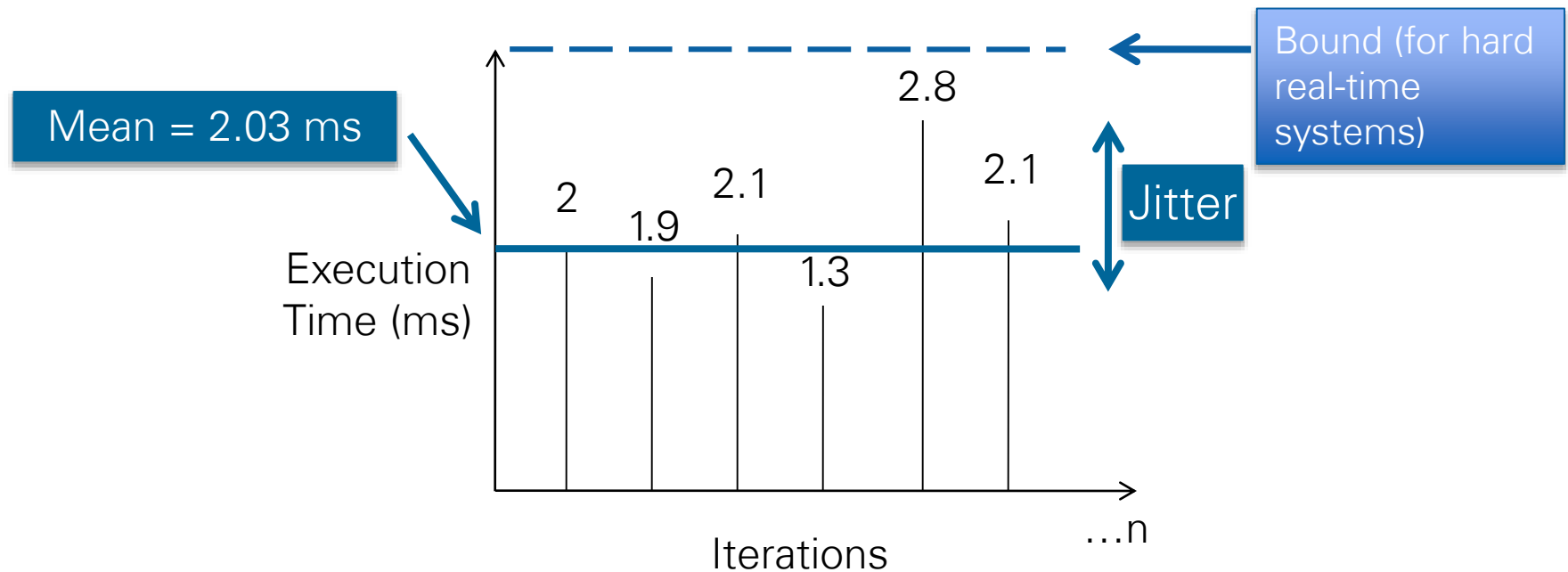
# Key Careabouts for Critical Applications

- **Jitter:** execution time variability of a given operation or application



# Key Careabouts for Critical Applications

- **Determinism:** a condition that is met if an operation or application has bounded jitter



# Running Real-Time Code – Modes

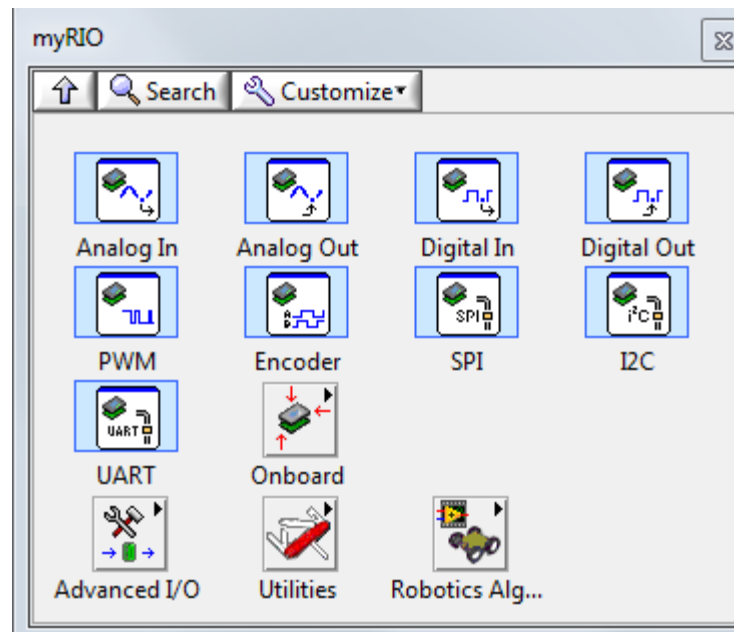
- “Real-Time Code” typically refers to VI’s running on a target that are at processor level (non-FPGA).
- RT code can be run in two modes.
  - Stand-alone
  - Interactive Front Panel Mode
- In both modes, the code is deployed to the target (NI myRIO) and is running on the NI myRIO processor.





# NI myRIO Express VIs - Overview

- **Express VIs** are special VIs that allow the user to configure their settings when placed on the block diagram.
  - This avoids having to wire controls/constants to terminals taking up time and block diagram space.
- NI myRIO has a palette of express VI that can be found by **right-clicking** on the block diagram and navigating to **myRIO**.



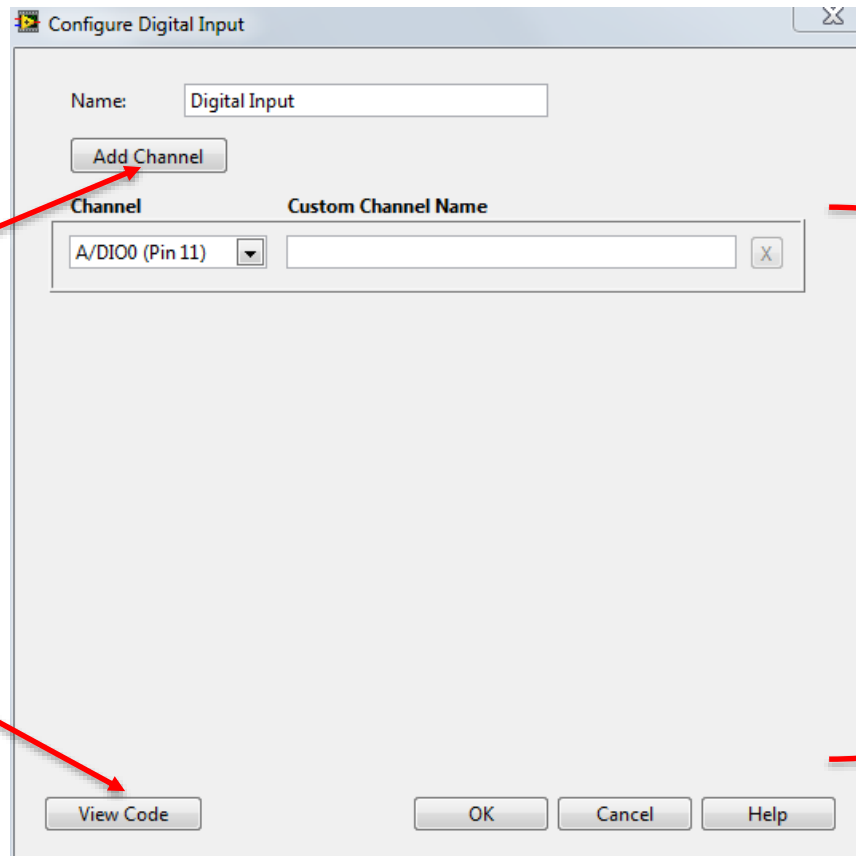
# NI myRIO Express VIs - Configuration

- When an Express VI is placed on the block diagram, a configuration dialog box opens.

This is the Digital Input Express VI.

More I/O channels can be accessed by selecting **Add Channel**.

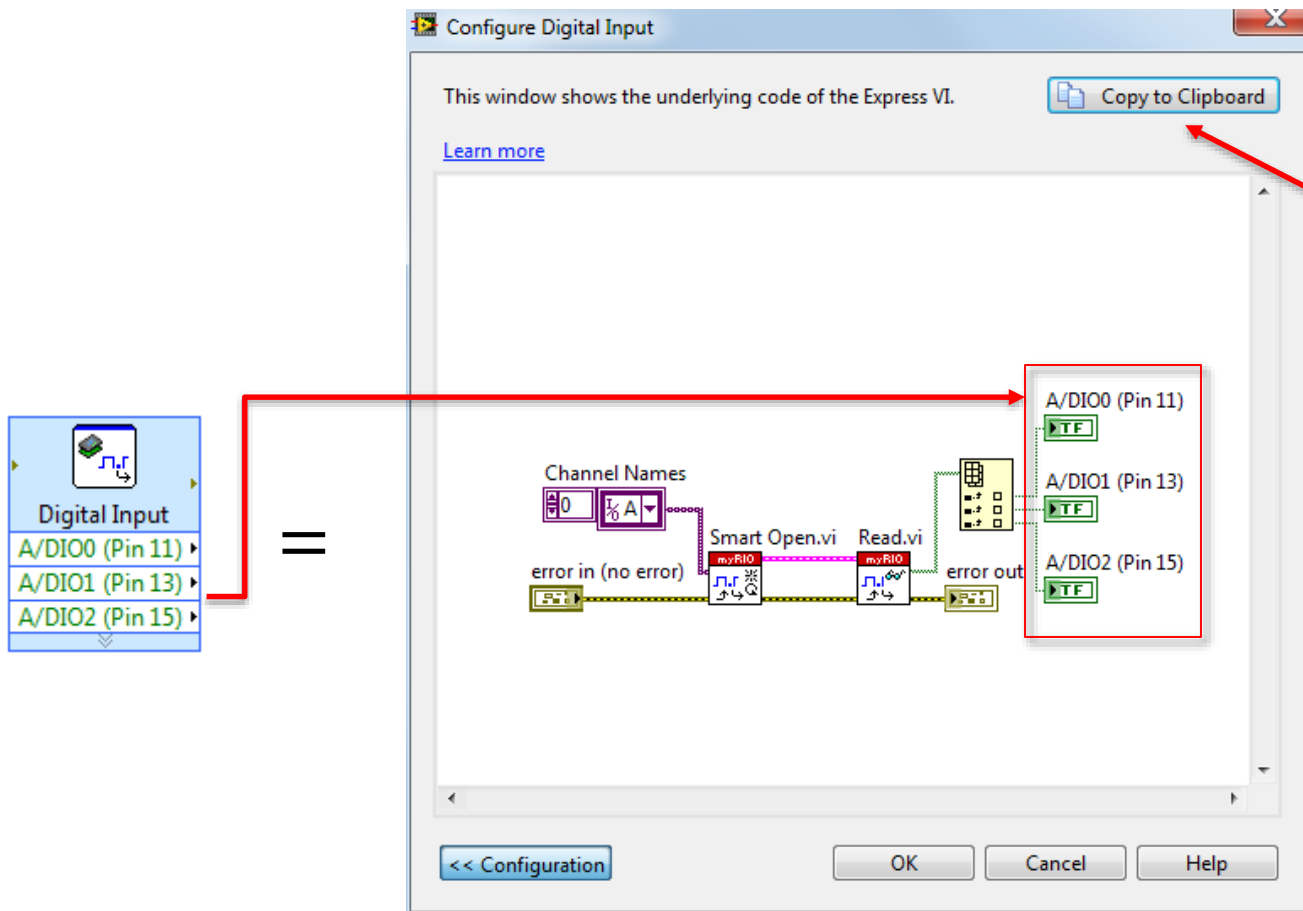
**View Code** allows you to inspect the underlying G code that the Express VI generates



Relevant configuration options are shown in this space.

# NI myRIO Express VIs – G-Code

- If the **View Code** button is selected in the configuration dialog, LabVIEW will generate a window like this:



Copy to Clipboard allows the user to copy this code into their VI if they want to customize options programmatically.

# Design with Familiar Models of Computation



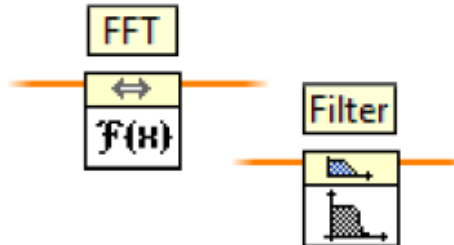
NATIONAL INSTRUMENTS

# LabVIEW™

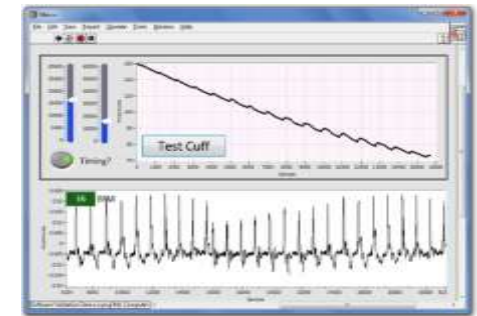
A Highly Productive Graphical Development Environment for Scientists and Engineers



Hardware APIs



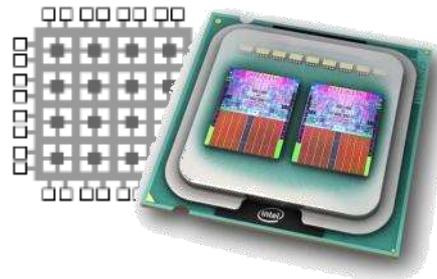
Analysis Libraries



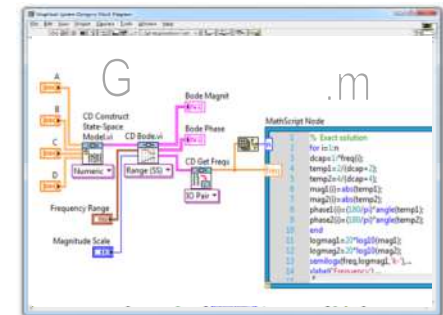
Custom User Interfaces



Deployment Targets



Technology Abstractions



Programming Approaches

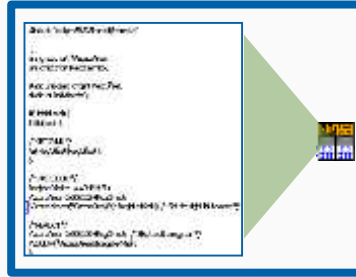


# Models of Computation

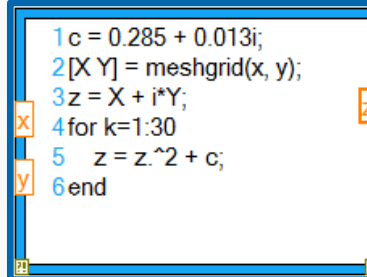
Data Flow



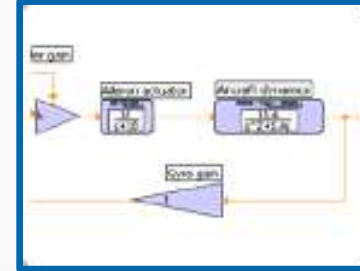
C Code



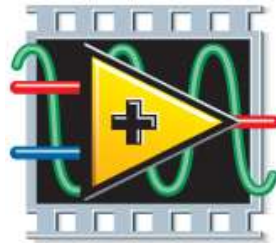
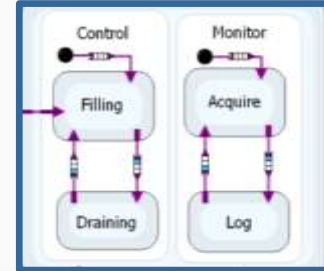
Textual Math



Simulation



Statechart



NATIONAL INSTRUMENTS

# LabVIEW™

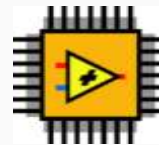
Desktop



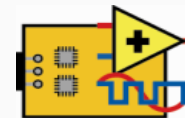
Real-Time



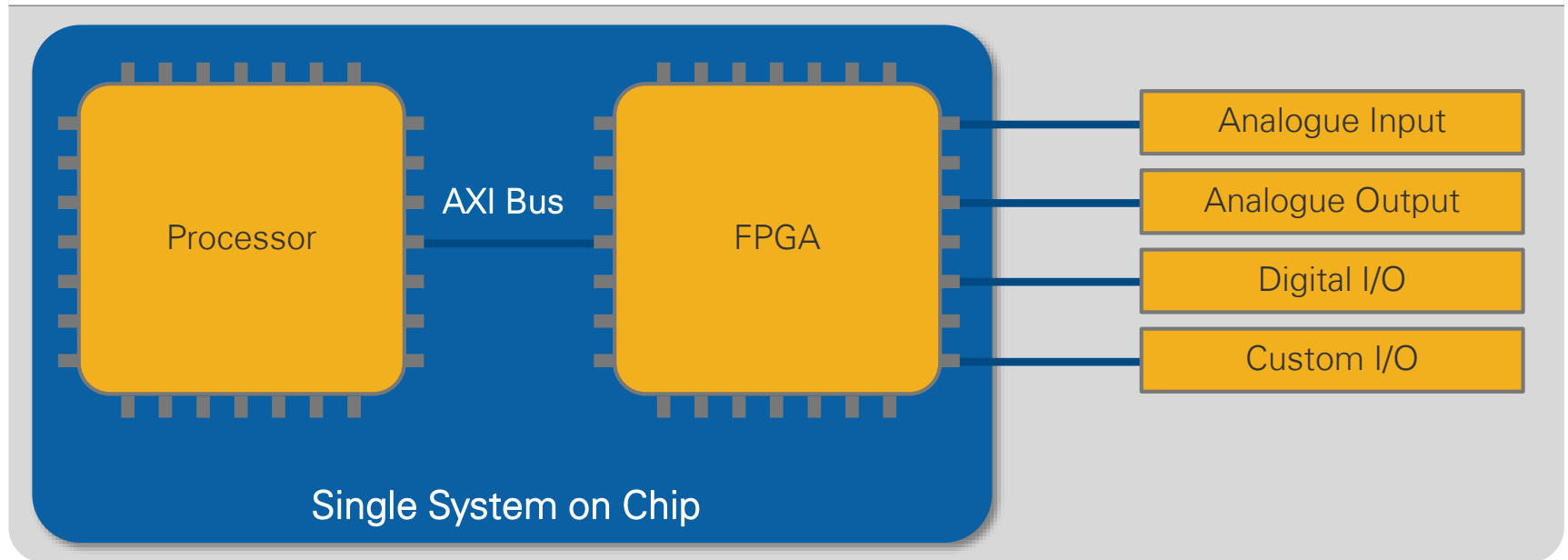
FPGA



Microprocessors



# myRIO: what is the operating system?



# NI Linux Real-Time



- Unlock the vast Linux **ecosystem**



## Database

Raima  
MySQL  
SQLite  
MongoDB  
CouchDB



## Security

OpenVPN  
IP Tables  
System Logging  
fail2ban  
denyhost



## Code Reuse

C/C++  
Shell Scripting  
Python  
Ruby  
Perl



## Connectivity

Isshd  
IPv6  
SNMP  
NTP  
netstat

# NI Linux Real-Time



- **Reuse** C/C++ code in and alongside Real-Time applications built with LabVIEW



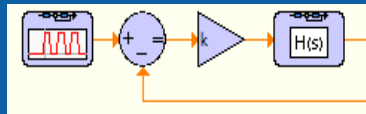
- Program the processor in C/C++ using Eclipse IDE
- A custom distribution of Eclipse is available for FREE download from [ni.com](http://ni.com)
- Examples for NI myRIO provided with download

# Deploy to Hardware Through LabVIEW

MathScript RT  
Module

```
1 A = [0 1; -2 -4];  
2 eigs = eig(A);  
3 detA = det(A);
```

Control Design &  
Simulation Module



The MathWorks Inc.  
software development  
environment

Your .m code

MATLAB®

Your .mdl code

Simulink®

Simulink Coder™

MATLAB® and Simulink® are registered trademarks  
of The MathWorks, Inc.

LabVIEW Real-Time

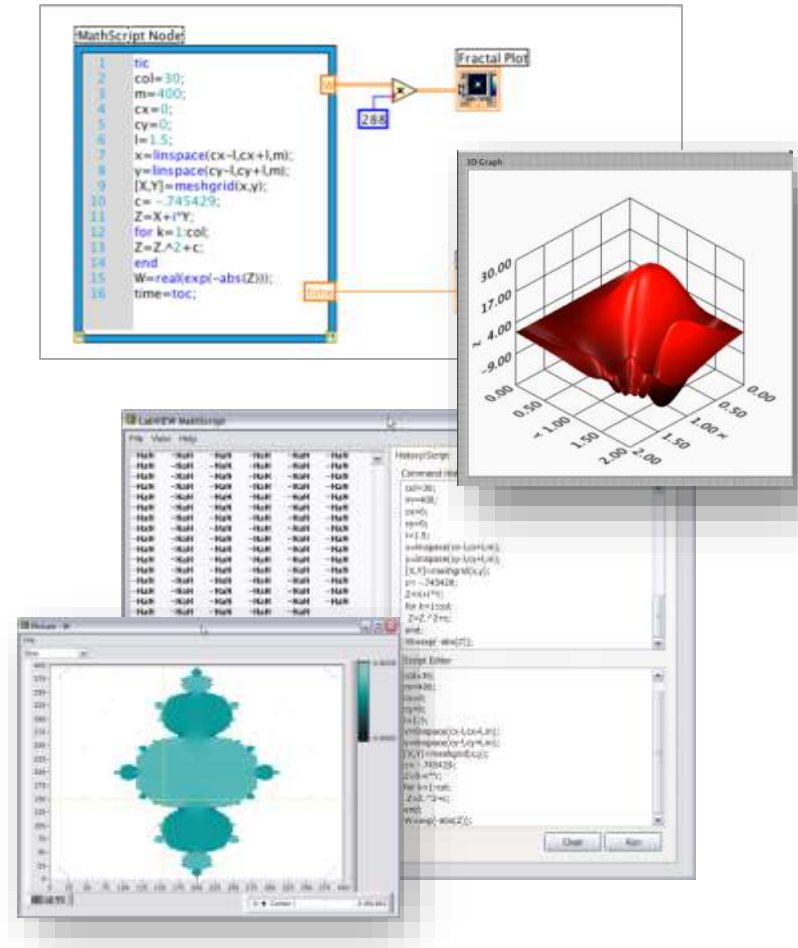
CompactRIO, Single-Board  
RIO, myRIO, PXI or desktop





# LabVIEW MathScript RT Module

- Text-based controls, signal processing, analysis, and math
  - 900 built-in functions / user-defined functions
  - Reuse many of your .m file scripts created with The MathWorks, Inc. MATLAB® software and others
  - Based on original math from NI MATRIXx software
- A native LabVIEW solution
  - Interactive and programmatic interfaces
  - Does not require 3<sup>rd</sup>-party software
  - Enables hybrid programming
  - Works w/ LabVIEW Real-Time



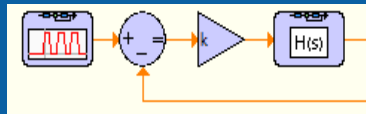
MATLAB® is a registered trademark of The MathWorks, Inc..

# Deploy to Hardware Through LabVIEW

MathScript RT Module

```
1 A = [0 1; -2 -4];  
2 eigs = eig(A);  
3 detA = det(A);
```

Control Design & Simulation Module



The MathWorks Inc. software development environment

MATLAB®

Your .m code

Simulink®

Your .mdl code

Simulink Coder™

MATLAB® and Simulink® are registered trademarks of The MathWorks, Inc.

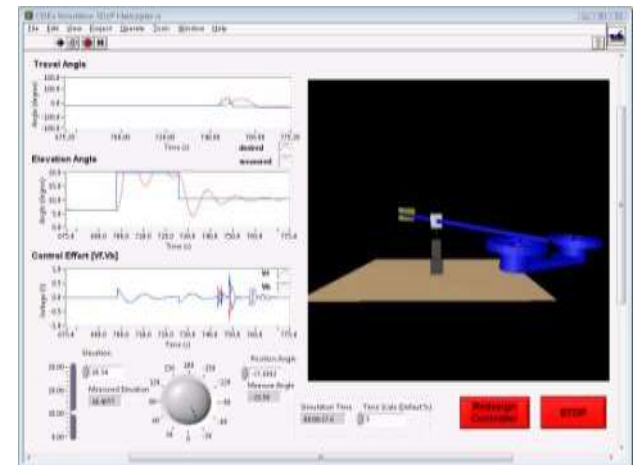
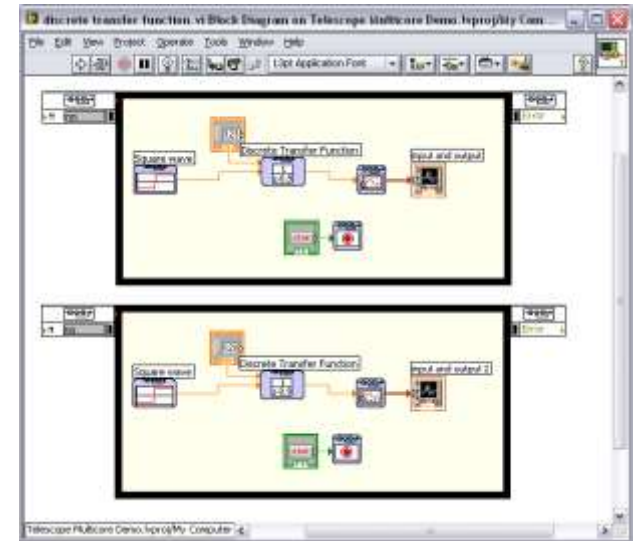
LabVIEW Real-Time

CompactRIO, Single-Board RIO, myRIO, PXI or desktop



# LabVIEW Control Design & Simulation

- Complete simulation and real-time implementation capability - stay in one environment from design to test to implementation
- Easily create parallel and multirate simulation or control loops, *leverage multicore*
- Custom user interface to change and observe parameters as simulation or control system is running
- Use VIs or programming structures inside or outside of simulation loops:
  - Integrated design and simulation, batch simulation
  - DAQ, RIO, Vision, or CAN for I/O



# Your Turn.

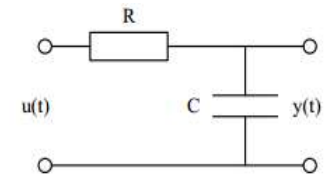
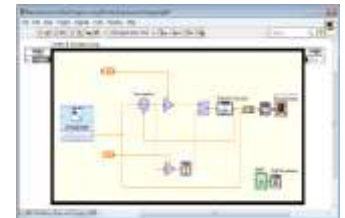
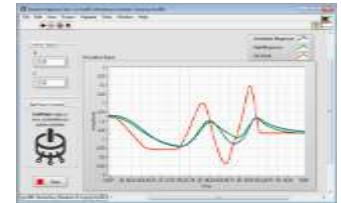
## Exercise

### Exercise 3

#### Control System Design with NI myRIO

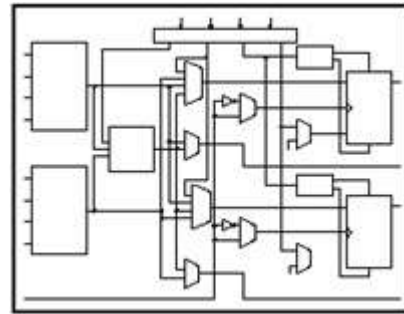
We have provided a simple PI control algorithm. You will...

- [20mins]** Create a transfer function to simulate a simple 1<sup>st</sup> order plant (RC circuit)
- [5mins]** Deploy the control algorithm and transfer function to myRIO
- [30mins]** Move from simulation to the real world, to control a physical RC circuit

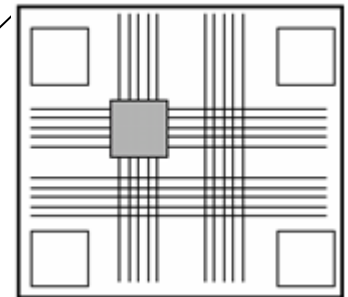
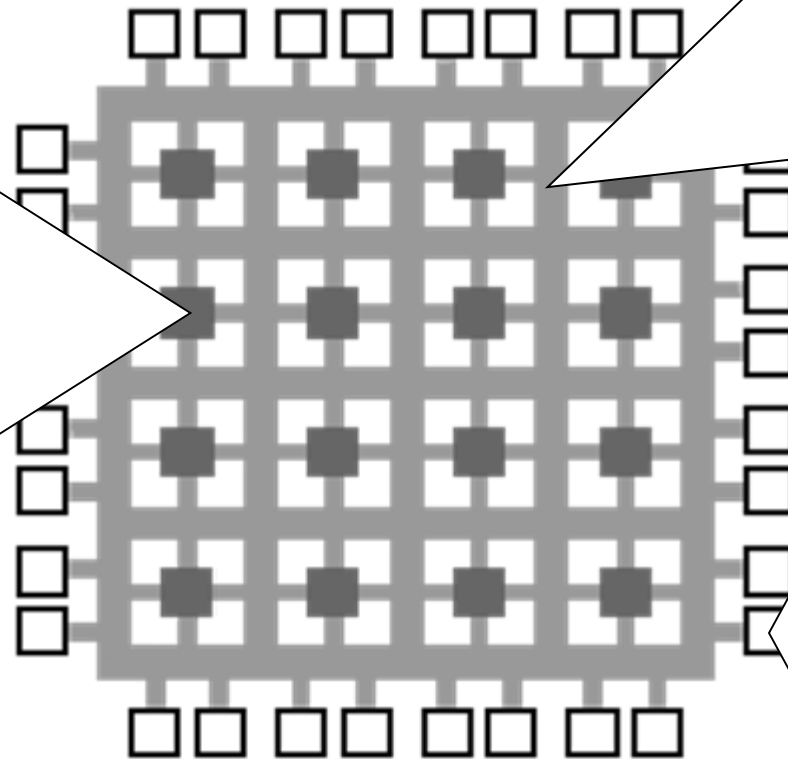


# Introduction to LabVIEW FPGA

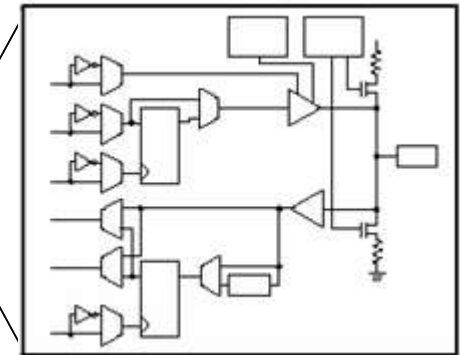
# FPGA Technology



**Logic  
Blocks**



**Programmable  
Interconnects**



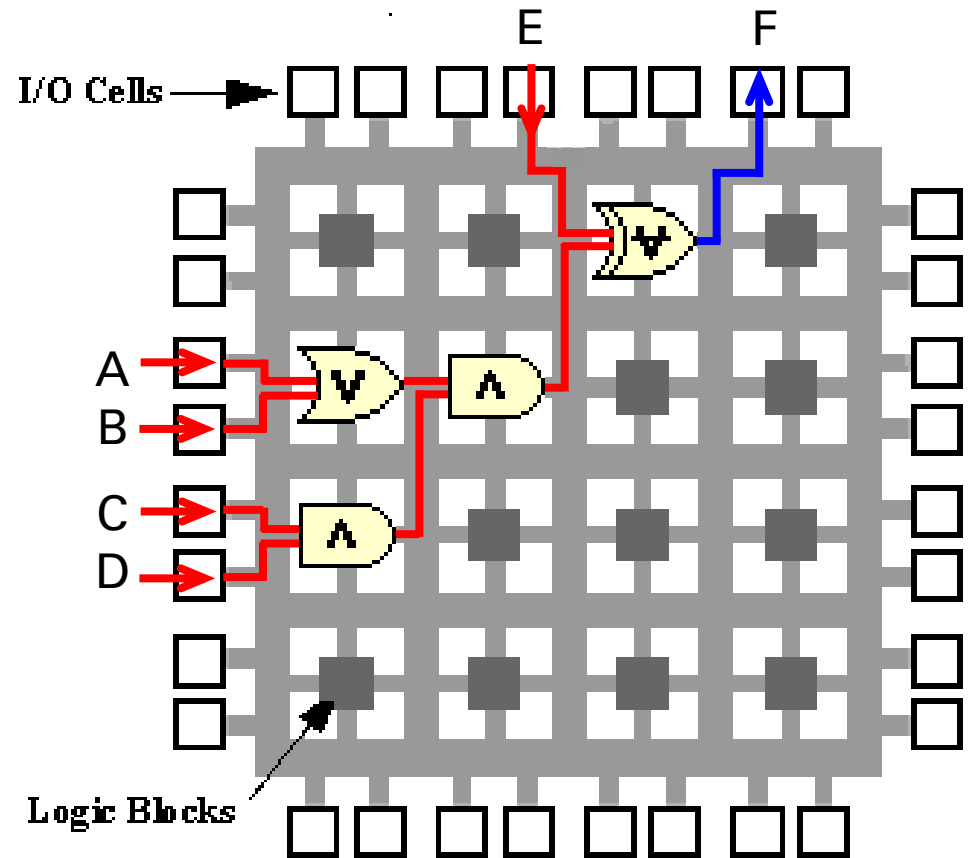
**I/O Blocks**



# FPGAs are Dataflow Systems

Implementing  
Logic on FPGA:

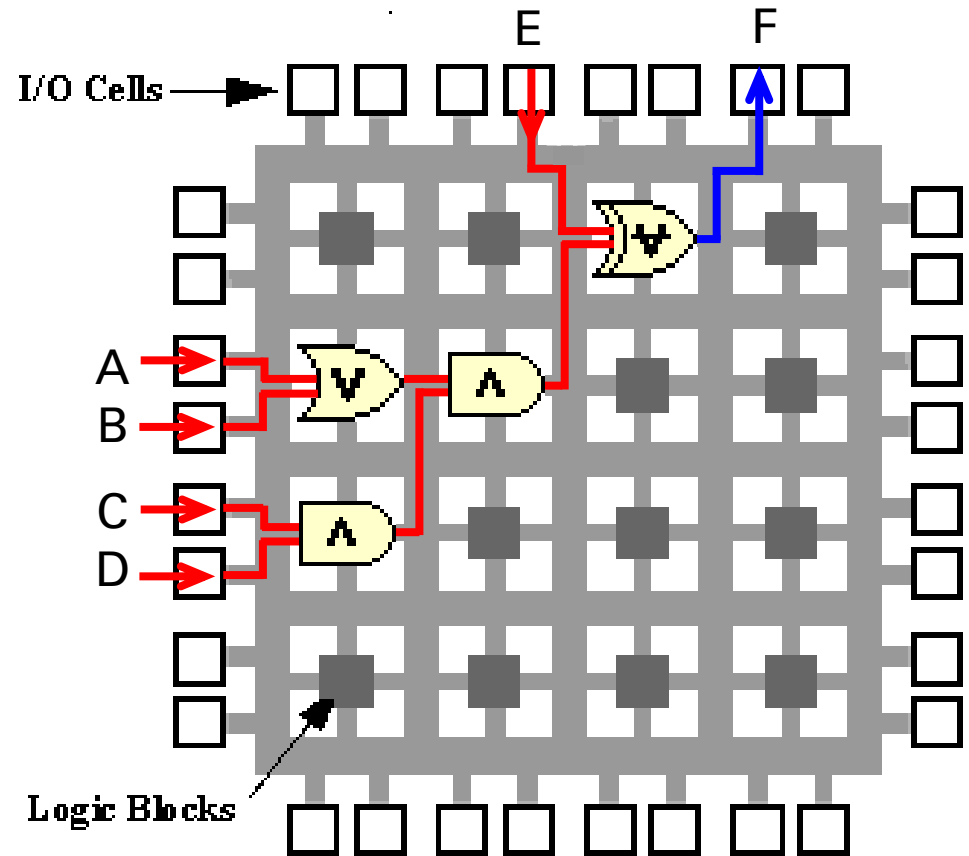
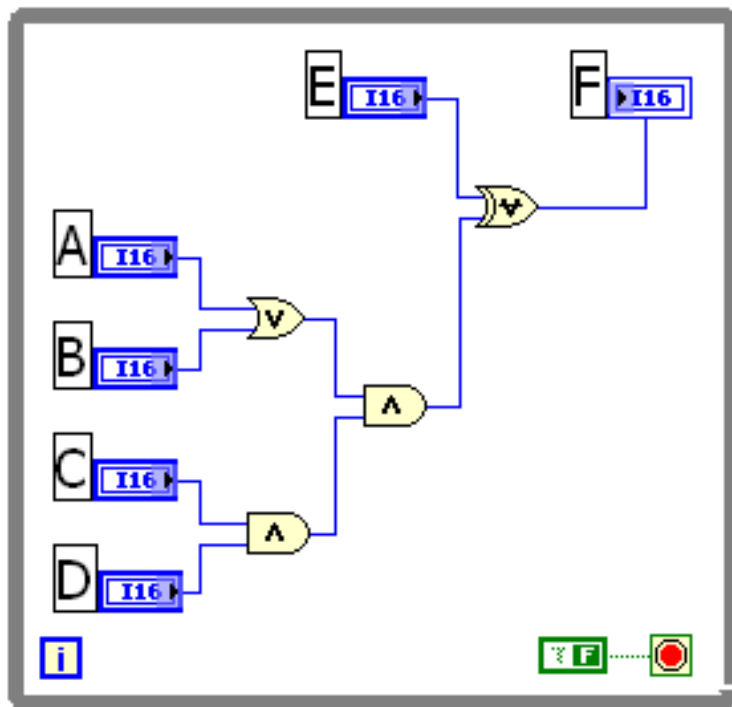
$$F = \{(A+B)CD\} \oplus E$$

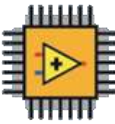


# FPGAs are Dataflow Systems

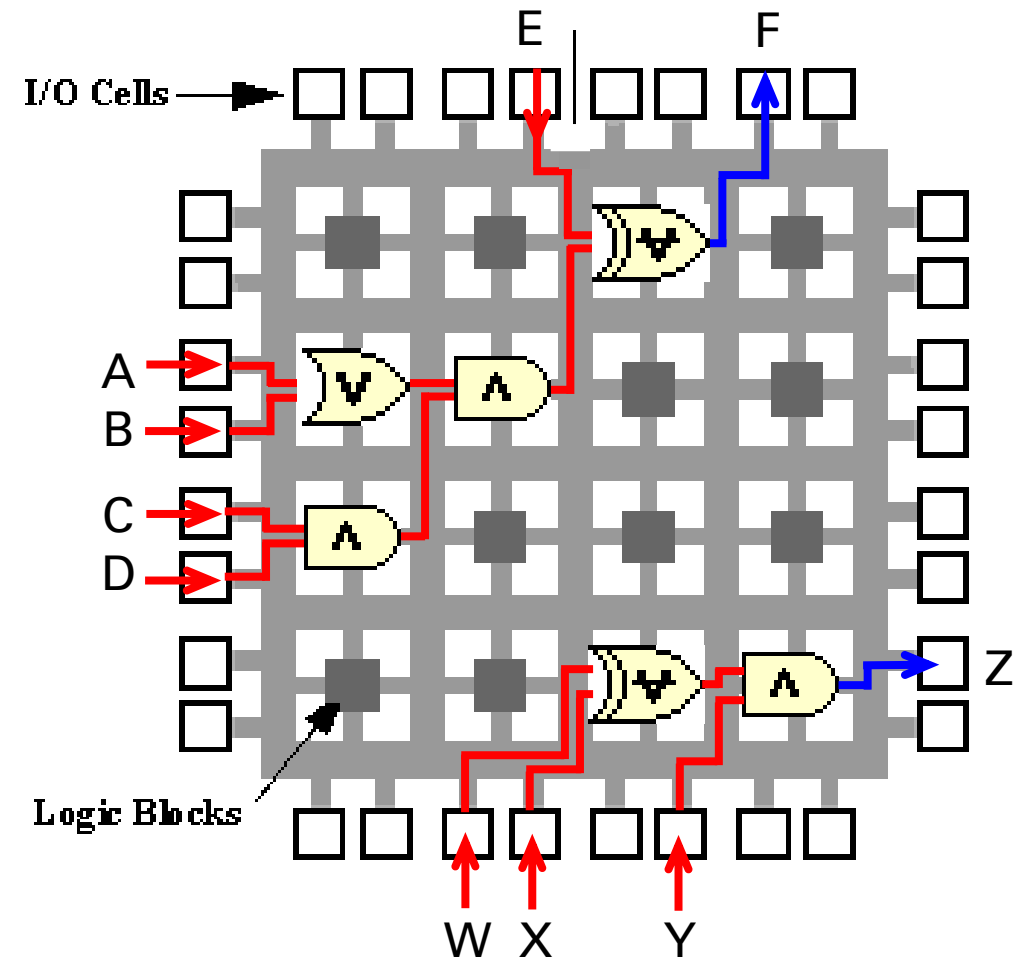
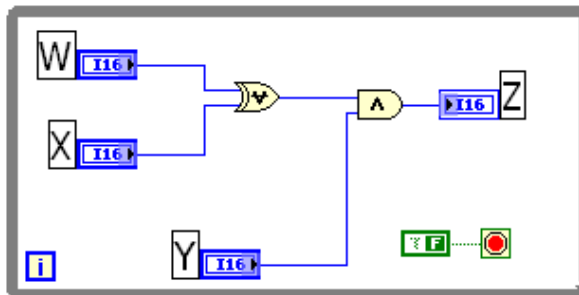
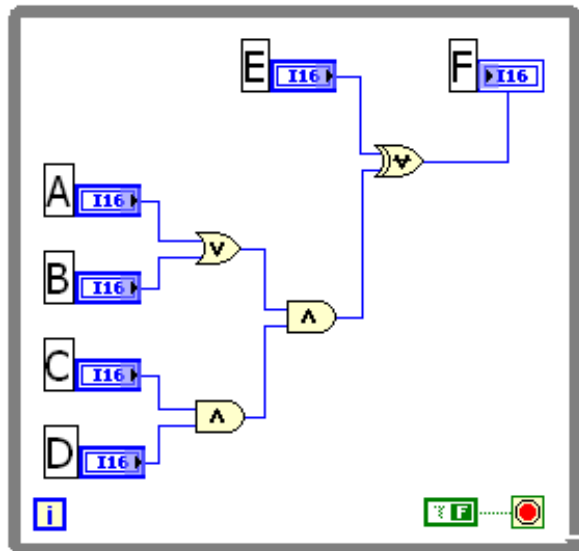
Implementing Logic on FPGA:  $F = \{(A+B)CD\} \oplus E$

LabVIEW FPGA Code

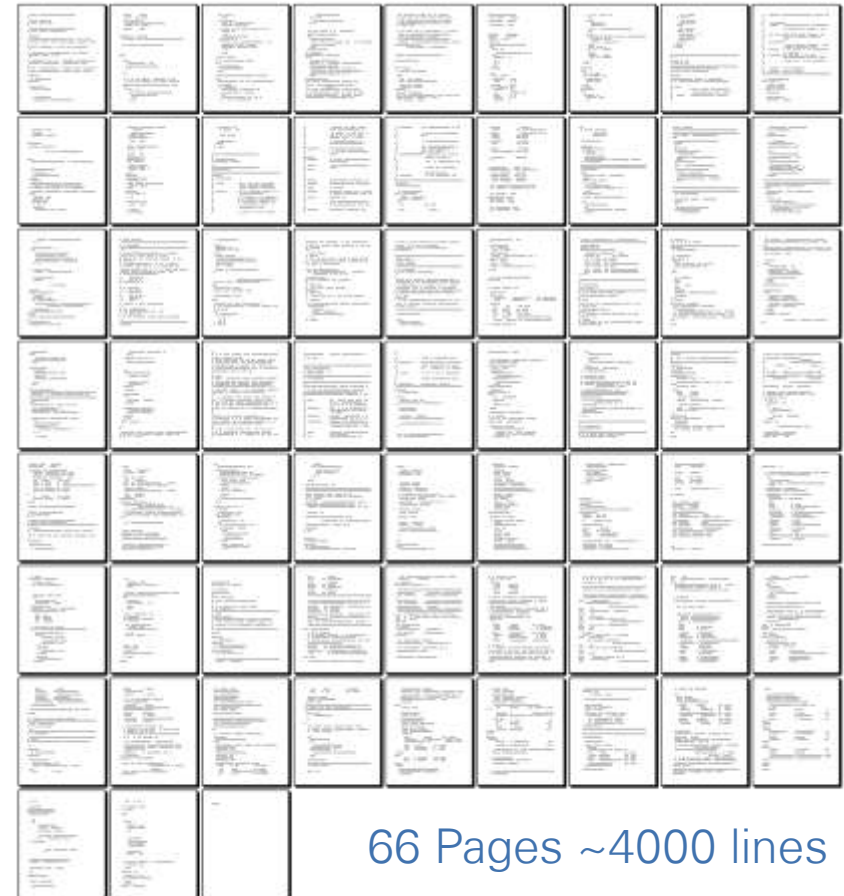
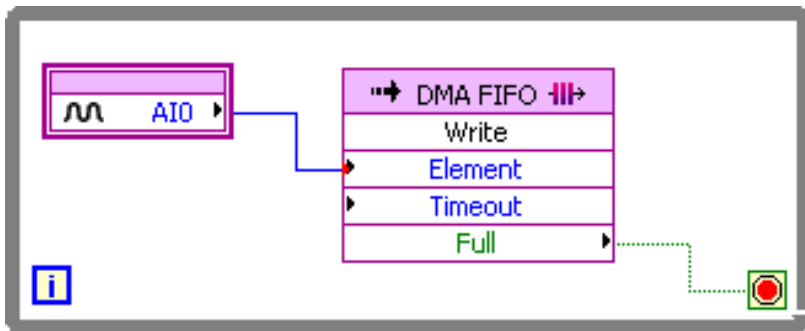
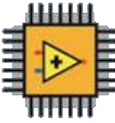




# FPGAs are Parallel Dataflow Systems

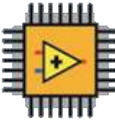


# LabVIEW FPGA vs. VHDL



66 Pages ~4000 lines

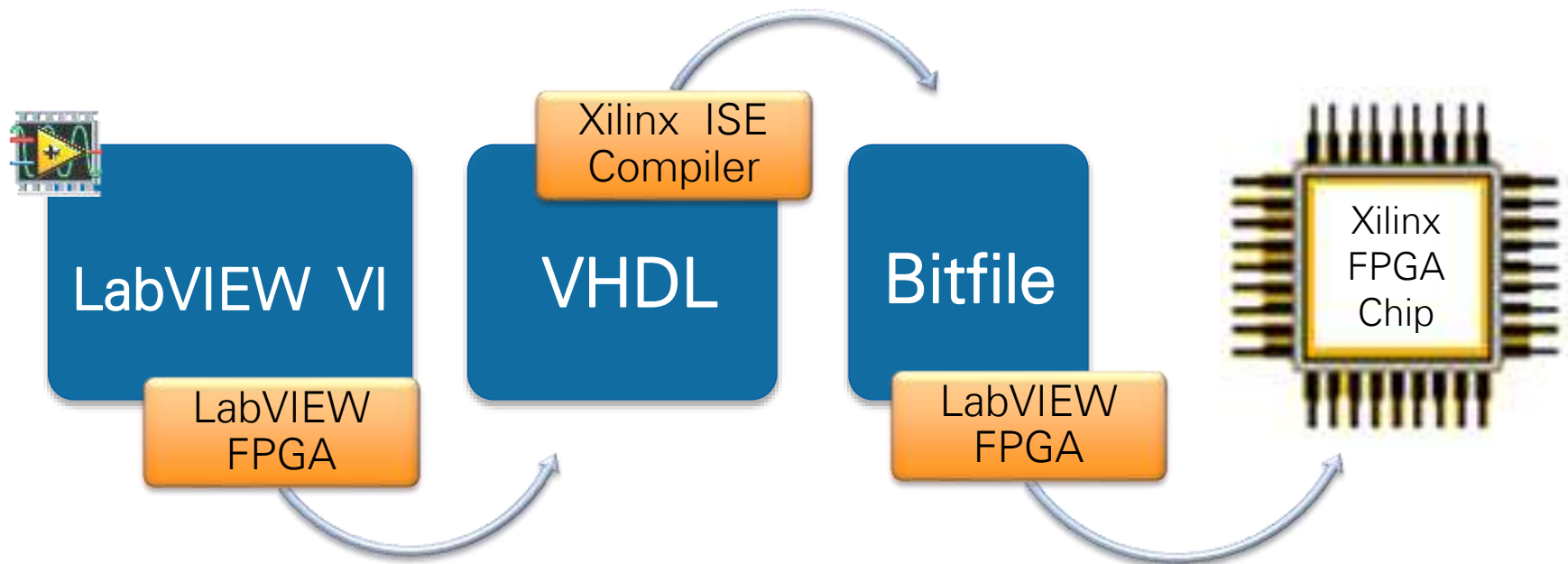
## I/O with DMA



# Why Are FPGAs Useful?

- **True Parallelism**  
Provides parallel tasks and pipelining
- **High Reliability**  
Designs become a custom circuit
- **High Determinism**  
Runs algorithms at deterministic rates down to 25 ns (faster in many cases)
- **Reconfigurable**  
Create new and alter existing task-specific personalities

# LabVIEW FPGA: How does it work?



# Resources and Next Steps



# NI myRIO Kits | [ni.com/myrio](http://ni.com/myrio)



## Starter

- LEDs & switches
- 7-segment display
- Potentiometer
- Thermistor
- Photo resistor
- Hall effect
- Microphone/Speaker
- Battery holder
- DC motor



## Mechatronics

- DC gear motors/encoders
- H-bridge driver
- Accelerometer
- Triple-axis gyro
- Infrared proximity sensor
- Ambient light sensor
- Ultrasonic range finder
- Compass
- Hobby servo motors



## Embedded

- RFID reader kit
- Numeric keypad
- LED matrix
- Digital potentiometer
- Character LCD
- Digital temp sensor
- EEPROM

# NI myRIO | Courseware



## 2 Discrete LED

LEDs, or light-emitting diodes, provide simple yet essential visual indicators for system status and error conditions. Figure 2.1 shows the four types of LEDs included in the SparkFun "LED Misad Bag (5mm)" kit (<http://www.sparkfun.com/products/9881>).



**Learning Objectives:** In this module you will create a standard interface circuit to verify correct operation of the LED, learn interface circuit design principles and related LabVIEW programming techniques, make some basic modifications to extend your understanding of the interface, and then challenge yourself to design a system that integrates the discrete LED with additional components or devices.

### 2.1 Component Verification

Follow these steps to verify correct operation of the discrete LED component.

Select these parts:

- Resistor, 220 ohm
- "Basic Red" LED from SparkFun 0881
- Breadboard
- Connecting wires (need details)

**Download the LabVIEW project:** Download the project Discrete LED demo.lvproj from need details.

### 2.2 BASIC MODIFICATIONS

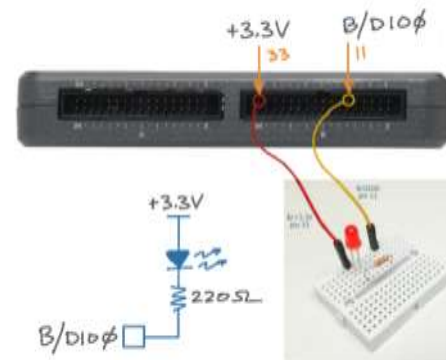


Figure 2.2: Discrete LED verification circuit: schematic diagram, recommended breadboard layout, and connection to NI myRIO MXP Connector II.

# C Support for NI myRIO – ni.com/myrio/c-support



**NATIONAL INSTRUMENTS**

Contact Us | United States

NIWeek

Products & Services Solutions Support Developer Zone Academic Events Company

My Profile MyNI Notifications Parts List Cart

Home > Developer Zone > Tutorial > C Support for NI myRIO

## C Support for NI myRIO

Publish Date: Jun 18, 2013 | 0 Ratings | 0.00 out of 5 | PDF | Submit your review

### Overview

NI myRIO is an embedded hardware device designed specifically to help students design re... quickly and affordably than ever before. NI myRIO is based on NI reconfigurable I/O (RIO) tec... program both a processor running a real-time OS and a customizable FPGA. In addition to N... processor is fully programmable in C or C++ using the default shipping personality placed o... using only the LabVIEW FPGA Module. Follow the steps below to start programming the NI n...

1. Install the Eclipse integrated development environment (IDE) and compile tools.
2. Download and install the LabVIEW for myRIO Module. This module provides you with th... well as some software utilities, which offer useful information about your device.
3. Download the NI myRIO C Examples and Documentation. The NI myRIO C examples ar... supported if you use the tools provided in Step 1.
4. Connect power to the NI myRIO device.
5. Connect the USB cable from NI myRIO to your development computer.
6. The **NI myRIO USB Monitor** appears and displays the IP address of the NI myRIO device



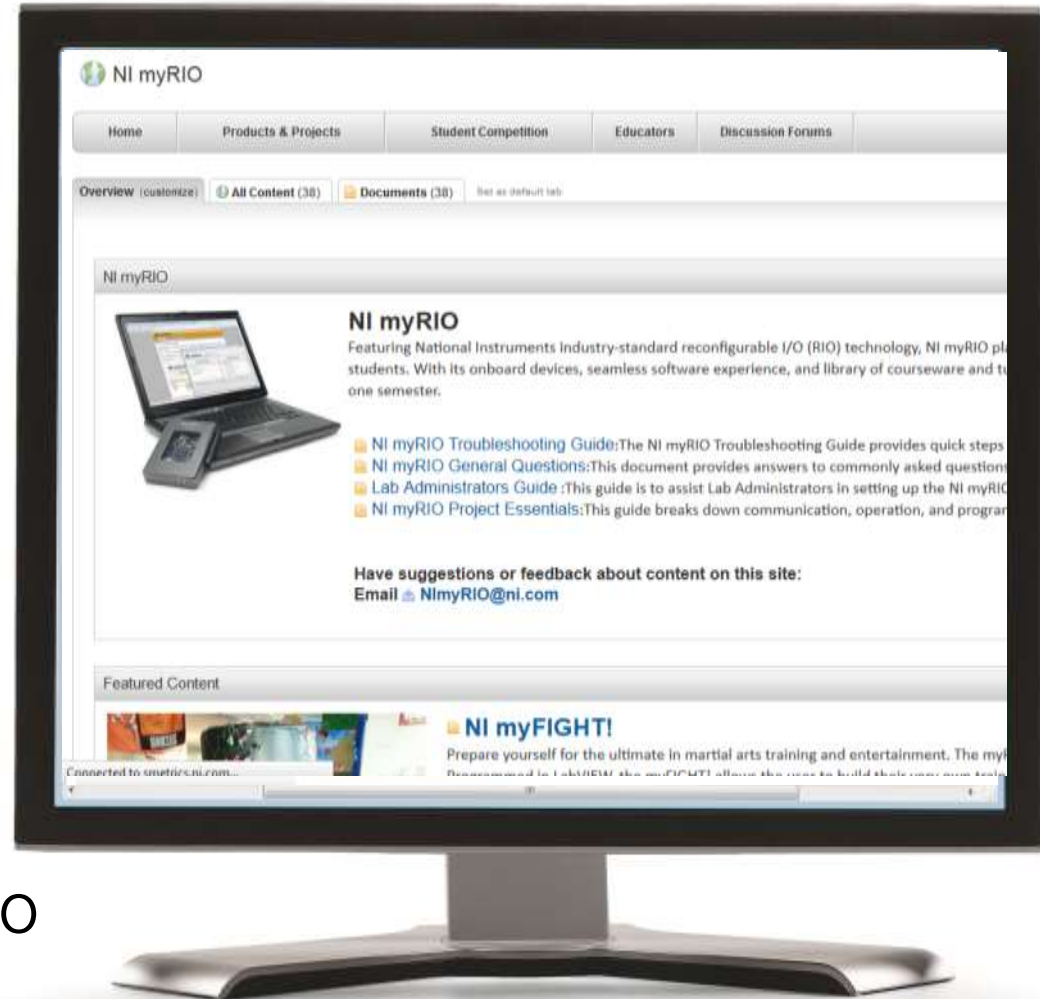
- Processor Programming
- Eclipse IDE
- C/C++ Support
- Free
- One Installer from ni.com

# Learn More About Programming NI myRIO



ni.com/learn-myrio

# Learn More About Programming NI myRIO



ni.com/community/myrio

